

missing topics

→ Crystal Process model / Crystal Agile Process model

→ Crystal method is an agile software development approach that focuses primarily on people & their interaction rather than on process or tools.

2 assumptions

- Teams will find ways to improve
- Every project is unique
- Generally they are characterized by colors, according to the number of people involved in the project.

Green → 8 or fewer

Yellow → 10 to 20

Orange → 20 to 50

Red → 50 to 100

6 people
clear

20 people
yellow

40 people
orange

80 people
red

more people
darker

→ Characteristics:

Adaptive:

Crystal is stretch to fit methodology meaning that processes & tools are not fixed, but have to be adjusted to meet requirements (at any time).

Ultralight:

Crystal doesn't involve too much documentation, management & reporting. As it maintains transparent workflow b/n team & client.

Human power:

This means that people involved in the project are vital & that the processes should be adapted to meet needs. (collaboration/interaction) are mandatory!

→ Properties

Frequent delivery: It allows you to frequently deliver, tested code/application to real users.

(every 2 months & min 2 deliveries per project).

Reflective improvement: no matter how bad or good the product is, there are always areas where the product can be improved.

Osmotic communication: with the team who work co located information flow around the team

This allow them to pick up valuable information even without being directly involved in the discussion.

Personal safety :- the only way to build a healthy working atmosphere & a true culture is by practicing open & honest communication.

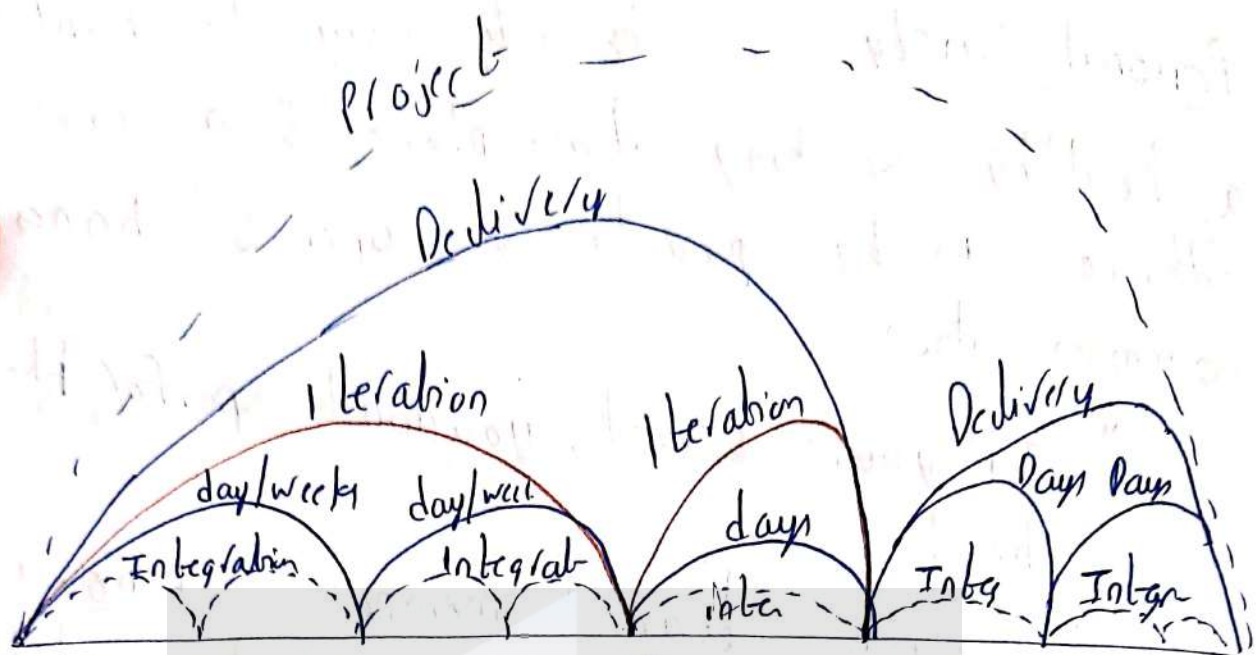
" If you're scared, you don't perform the best "

Focus: each team member knows exactly what to work on which enables them to focus their attention & avoid switching from one task to another

Easy access to expert users: Crystal enables your team to maintain communication & get regular feedback from real users

A technical environment with automated tests, configurations management & frequent

integration :- very special tools for software team where the emphasis is on continuous integration so that the errors could be caught within minutes

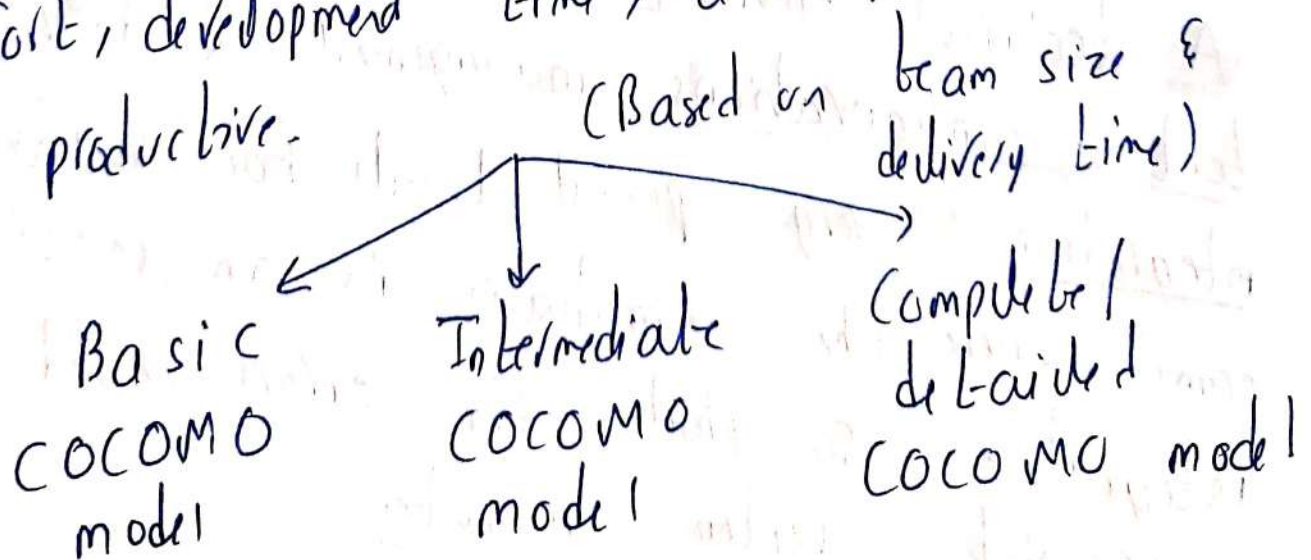


Process \nearrow
 name do explain so explain your

words

COCOMO model:

- Constructive Cost model
- It is used for calculate the effort, development time, average staff size & productive.



→ CoCoMo applied on 3 classes of s/w project

- i) Organic mode
- ii) Semi detached mode
- iii) Embedded.

kLOC
- kilo
lines
of code

| <u>project size</u> | <u>nature of project</u> | <u>innovation</u> | <u>deadline of project</u> |
|---------------------|--------------------------|-------------------|----------------------------|
|---------------------|--------------------------|-------------------|----------------------------|

i) organic

2 - 50 kLOC
(2 - 50000
lines of code)

small
size

ex: pay

roll,
inventory
project

low-high

not
tight

ii) Semi
detached
mode

50-300
kLOC

medium
size

ex: data

base
system

medium

medium

iii) Embedded

over
300 kLOC

large
size

ex: ATM,
air traffic
control

Signifi-
cant

Tight

ii) Basic CoCoMo model:

to calculate

$$E = a_b (KLOC)^{b_b}$$

$$D = c_b (E)^{d_b}$$

$$P = E/D$$

$E \leftarrow$ effort applied

$D \leftarrow$ development time

$P \leftarrow$ people required

a_b, b_b, c_b, d_b are constant co-efficient

based on classes

just for example

| Software project | a_b | b_b | c_b | d_b |
|------------------|-------|-------|-------|-------|
| Organic | 2.4 | 1.05 | 2.5 | 0.38 |
| Semide tailed | 3.0 | 1.12 | 2.5 | 0.35 |
| Embedded | 3.6 | 1.20 | 2.5 | 0.32 |

excl Given size = 400 LOC.

effort? development = ? For
all 3 models

$$E = ab(KLOC)^{bb}$$
$$D = (b(E))^{db}$$

organic model

$$E = 2.4(400)^{1.05}$$

$$= 1295.31 \text{ PM}$$

(Person month)

$$D = 2.5(1295.31)^{0.38}$$
$$= 38.07 \text{ PM}$$

embedded
semi-aided

$$E = 3.6(400)^{1.20}$$
$$= 4772.81 \text{ PM}$$

$$D = 2.5(4772.8)^{0.35}$$
$$= 38 \text{ PM}$$

semi-aided model

$$E = 3.0(400)^{1.12}$$
$$= 2462.79 \text{ PM}$$

$$D = 2.5(2462.79)^{0.35}$$
$$= 38.45 \text{ PM}$$

iii) Intermediate CoCoMo model:

- It is an extension of basic
- Set of 15 additional predictors

to estimate

Product attributes

- required software reliability (RELY)
- database size (Data)
- Product complexity (CPLX)

Personal attributes

- Analyst capability (A CAP)

- A.P.P experience (AEXP)

- Programmer capability (P CAP)

- Virtual

language experience (VEXP)

- Programmer language experience (LEXP)

Computer attributes

- Execution time constraints

- main storage constraints (STOR)

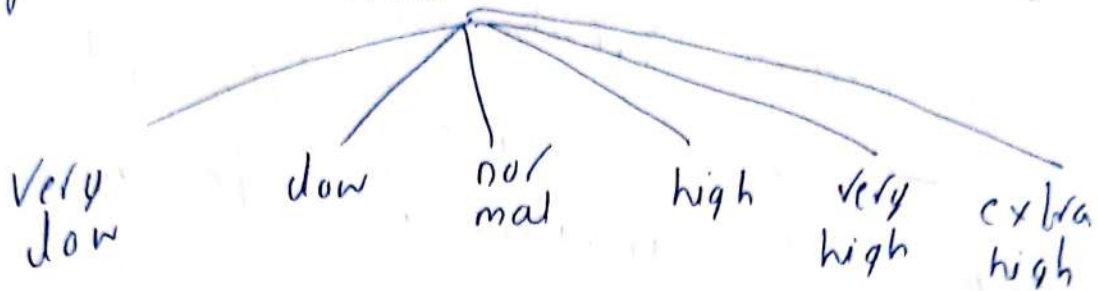
- virtual machine readability (VIRT)

- Computer turn around time (TURN)

④ Project Attributes

- Modern programming practices (MODP)
- Use of s/w tools (TOOL)
- required development schedule (SCED)

→ each cost driver is rated for the given project environment.



$$\Rightarrow E = a_i (KLOC)^{b_i} * EAF$$

$$D = C_i (E_i)^{d_i}$$

Effort adjustment factor.

(multiply all 15 values)
commonly given in question

| Project | a_i | b_i | C_i | d_i |
|---------------|-------|-------|-------|-------|
| organic | 3.2 | 1.05 | 2.5 | 0.38 |
| semi detached | 3.0 | 1.12 | 2.5 | 0.35 |
| embedded | 2.8 | 1.20 | 2.5 | 0.32 |

iii) Detailed / Complete CoCoMo model

all characteristics of intermediate with an assessment of the cost driver impact on each step of the software engineering.