ESTIMATION FOR SOFTWARE PROJECTS

Sample problems and solutions

Unit II

FAQs

It is given that the complexity weighting factors for I, O, E, F, and N are 4, 5, 4
0, and 7, respectively. It is also given that, out of fourteen value adjustment
ctors that influence the development effort, four factors are not applicable,
ach of the other four factors has value 3, and each of the remaining factors ha
alue 4. The computed value of the function point metric is _____. [GATE CS 201

(A) 612.06
(B) 404.66
(C) 305.09
(D) 806.9

Solution: Correct Answer is (A).

For example, if the software project has 30 external inputs, 60 external outputs, 23
external inquiries, 8 internal logical files, and 7 external interface files, and the valu
adjustment factors are 3, 3, 3, 3, 0, 0, 0, 0, 4, 4, 4, 4, 4, and 4, then the function point
metric can be calculated as follows:

UFP = (30 * 4) + (60 * 5) + (23 * 4) + (8 * 10) + (7 * 7) = 606

VAF = 0.65 + (0.01 * (3 + 3 + 3 + 3 + 0 + 0 + 0 + 0 + 4 + 4 + 4 + 4 + 4 + 4)) = 1.01

FP = UFP * VAF = 606 * 1.01 = 612.06

FP estimation

- **FP is estimated using the following expression**

$$FP = \text{count total} \times [0.65 + 0.01 \times \Sigma(F_i)]$$

where F_i $(i = 1$ to $14)$ = *value adjustment factors* (VAF) whic
calculated based on the responses to the following questions:

1. Does the system require reliable backup and recovery?

2. Are specialized data communications required to transfer information from the application?

FP estimation

5. Will the system run in an existing, heavily utilized operational environment?

6. Does the system require online data entry?

7. Does the online data entry require the input transaction to be built over multiple screens or operations?

8. Are the ILFs updated online?

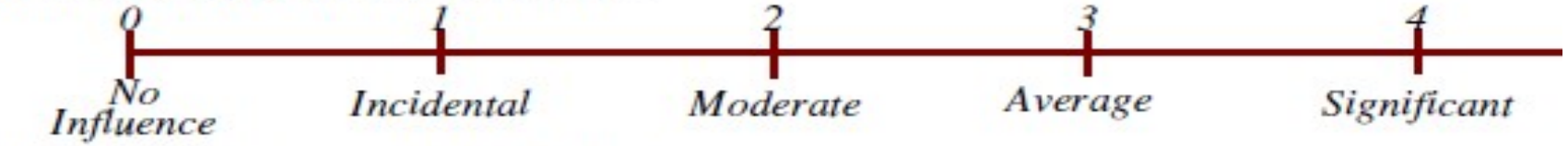
9. Are the inputs, outputs, files, or inquiries complex?

10. Is the internal processing complex?

11. Is the code designed to be reusable?

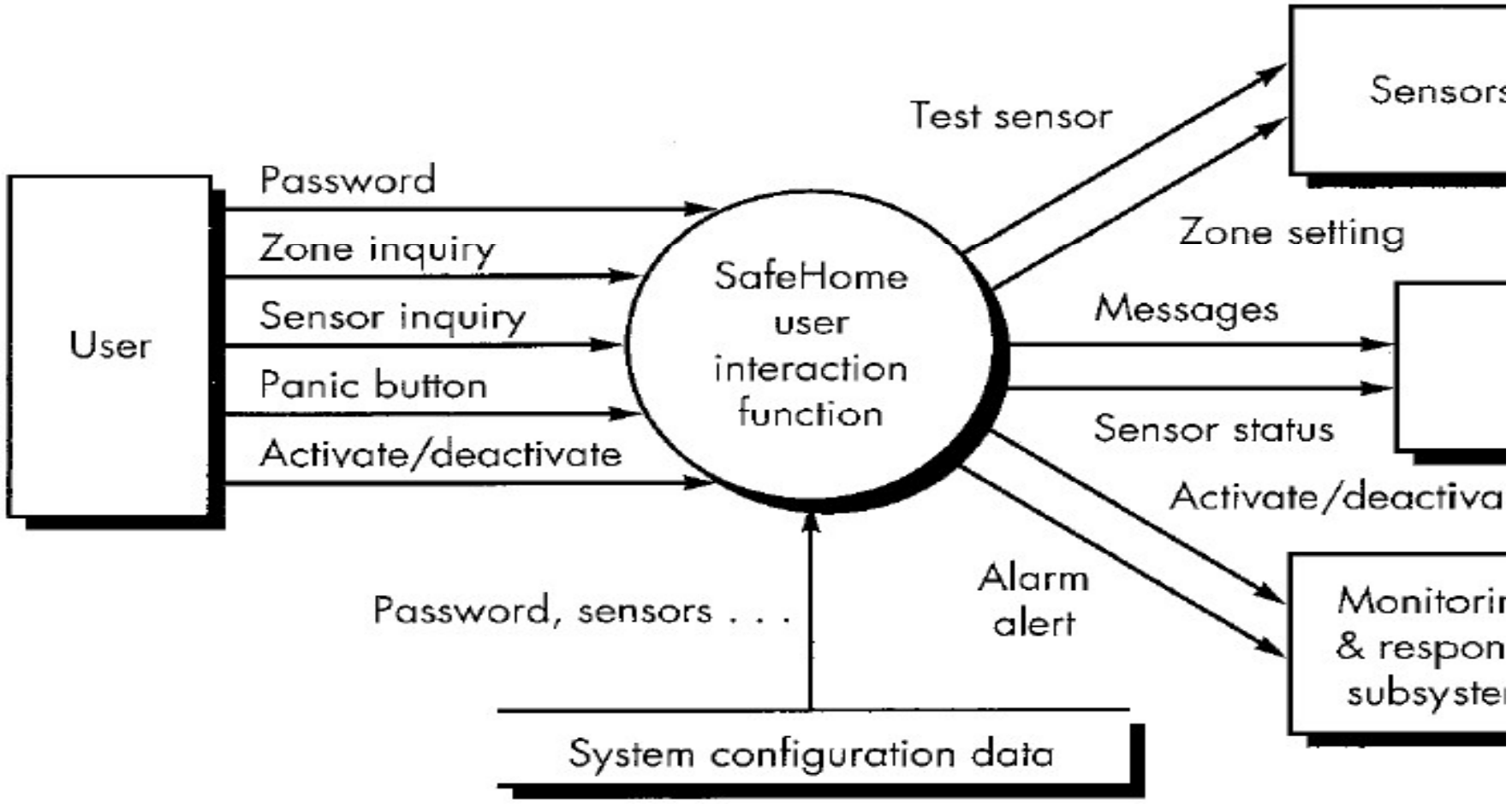
12. Are conversion and installation included in the design?

Rate each factor on a scale of 0 to 5.



FP Technique (Function Point Technique)

Example2: Part of analysis model for SafeHome software



Measurement parameter	Count		Simple	Average	Complex
Number of user inputs	3	×	3	4	6
Number of user outputs	2	×	4	5	7
Number of user inquiries	2	×	3	4	6
Number of files	1	×	7	10	15
Number of external interfaces	4	×	5	7	10
Count total					

+Assume Weighting Factor is simple:

+ Assuming that:
  - To complete one FP of work, the project requires 10 hours (Productivity = 10h
  - The total FP estimated is 200 FP for the project
  - The project team works 8 person-hours per working day
  - There are 20 working days in a month

+ Calculate the effort required to complete the sw project

Solution

To complete a project of 200 FP, you require:

+Effort in person-hours:

Effort = 200 FP * 10hrs/FP = **2000 person-hours**

+Effort in person-days:

Effort = 2000 hours/8 hours = **250 person-days**

+Effort in person-months:

Effort = 250 person-days/20 days = **12.5 person-months**

Example 3

Consider a project with the following functional units:

Number of user inputs	= 50
Number of user outputs	= 40
Number of user enquiries	= 35
Number of user files	= 06
Number of external interfaces	= 04

Consider Adjustment factor = 42

- Count = $50 \times 4 + 40 \times 5 + 35 \times 4 + 6 \times 10$

  $200 + 200 + 140 + 60 + 28 = 628$

- FP = 628 x(0.65+0.01(42)) $= 628 \times 1.07 = 6$

Example 4

An application has the following:

10 low external inputs, 12 high external outpu
internal logical files, 15 high external interfac
average external inquiries, and a value of

$$= 10 \times 3 + 12 \times 7 + 20 \times 7 + 15 + 10$$

$$= 30 + 84 + 140 + 150 + 48$$

$$= 452$$

$$= 452 \times 1.10 = 497.2.$$

Consider a project with the following paramet

(*i*) External Inputs:

(a) 10 with low complexity

(b) 15 with average complexity

(c) 17 with high complexity

(*ii*) External Outputs:

(a) 6 with low complexity

(b) 13 with high complexity

(*iii*) External Inquiries:

(a) 3 with low complexity

(*iv*) Internal logical files:

(a) 2 with average complexity

(b) 1 with high complexity

(v) External Interface files:

Other complexity adjustment factors are treated as aver the function points for the project.

Estimation with Usecases

Use cases provide a software team with insight into **software scope** and **requirements**.

Use cases are described using many **different formats and styles** - there is **no standard form**.

Use cases represent an **external view** (the user's view) of the software and can therefore be written at many **different levels of abstraction**.

Use cases **do not address the complexity of the functions** and features that are described.

Use cases **can describe complex behavior** (e.g., interactions) that involve many functions and features.

One person's **"use case"** may require months of effort while another person's use case may be implemented in a day or two.

Smith argues that any level of this structural hierarchy can be described by no more than 10 use cases.

Use cases that describe a large system are written at a much higher level of abstraction than

- use cases that are written to describe a single subsystem

Before use cases can be used for estimation,

- level within the structural hierarchy is established
- the average length (in pages) of each use case is determined
- type of software (e.g., real-time, business, engineering/scientific, WebApp, embedded) is defined
- Rough architecture for the system is considered.

relationship:[10]

$$\text{LOC estimate} = N \times \text{LOC}_{\text{avg}} + [(S_a/S_h - 1) + (P_a/P_h - 1)] \times$$

where

N = actual number of use cases

LOC_{avg} = historical average LOC per use case for this

$\text{LOC}_{\text{adjust}}$ = represents an adjustment based on n percer
is defined locally and represents the diffe
project and "average" projects

S_n = actual scenarios per use case

Use-Case Based Estimation

CAD software is composed of **three subsystem** groups

user interface subsystem

engineering subsystem group

infrastructure subsystem group

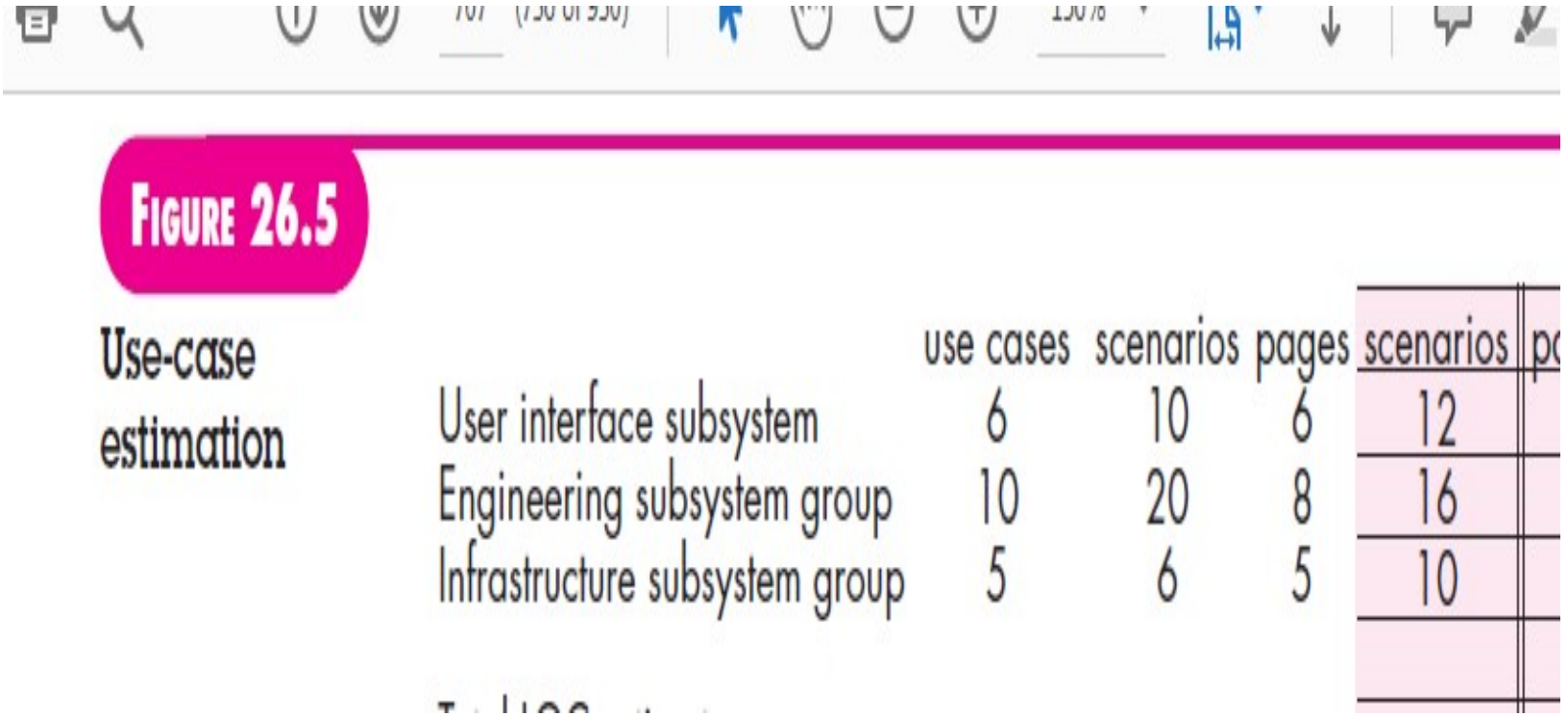


FIGURE 26.5

Use-case estimation

	use cases	scenarios	pages	scenarios	pa
User interface subsystem	6	10	6	12	
Engineering subsystem group	10	20	8	16	
Infrastructure subsystem group	5	6	5	10	

Reconciling Estimates

- Single estimate of project.
- Project duration (or) Cost.
- Total estimated effort for the CAD software ranges from a low of 46 person-months to a high of 68 person-months.
- Average estimate is 56 person-months.

TOPIC – 4
Empirical Estimation Techniques

Empirical Estimation Models

- An estimation model for computer software uses **empirically derived formulas** to **predict effort as a function of LOC** or **FP**.

- The empirical data that support most estimation models are derived from a limited sample of projects.

- The model should be **tested by applying data collected from completed projects**.

- **Plugging the data into the model**.

- Then **comparing actual to predicted results.**

- If agreement is poor, the model must be tuned and retested before it can be used.

The Structure of Estimation Models

- A typical estimation model is **derived using regression analysis on data collected** from past software projects.

- Overall **structure of such models** takes the form.

A typical estimla

past software pr

- where,
  - A,B and C are empirically derived constants.
  - E is effort in person-months
  - e_v is the estimation variable either **LOC (or) FP**

- LOC-Oriented estimation models in the literature as follows :

(e.g., problem complexity, staff experience, development

many LOC-oriented estimation models proposed in the lit

$E = 5.2 \times (\text{KLOC})^{0.91}$	Walston-Felix
$E = 5.5 + 0.73 \times (\text{KLOC})^{1.16}$	Bailey-Basili

- FP-Oriented estimation models in the literature as follows :

FP-oriented models have also been proposed. These inclu

$E = -91.4 + 0.355\ \text{FP}$	Albrecht and Ga
$E = -37 + 0.96\ \text{FP}$	Kemerer model

COCOMO-II Model

COnstructive **CO**st **MO**del (**COCOMO**) II is actually a hierarchy of estimation models that address the following areas.

- **Application Composition Model.**
  - Used **during the early stages of software engineering**
  - When **prototyping of user interfaces**
  - Consideration of **Software and System Interaction**
  - Assessment of **Performance**.
- **Early Design Stage Model.**
  - Used **once requirements have been stabilized** and
  - Basic **software architecture has been established**.
- **Post-Architecture-Stage Model.**
  - Used **during the construction of the software**.

- Like all estimation models for software, **COCOMO II** model require sizing information.
- **Three different sizing options** are available.
  - **Object Point**
  - **Function Point and**
  - **Lines of Source Code**
-

Object type	Complexity	
	Simple	Mediu
Screen	1	2
Report	2	5

- **Object Point**
  - Indirect software measure that is computed using counts of the number of,
    - Screens, Reports and Components likely to be required to build the application

- **Object point count** is then determined by **multiplying the original number of object instances by the weighting factor**.
- When component based development or general software re

the percent of reuse (%reuse) is estimated and the object poin

$$NOP = (\text{object points}) \times [(100 - \%\text{reuse})/100]$$

where NOP is defined as new object points.

To derive an estimate of effort based on the computed NOP rate" must be derived. Figure 26.7 presents the productivity rat

$$PROD = \frac{NOP}{\text{person-month}}$$

for different levels of developer experience and development e

Once the productivity rate has been determined, an estima

FIGURE 26.7 Productivity rate for object points.

Source: [Boe96].

Developer's experience/capability	Very low	Low	Nominal
Environment maturity/capability	Very low	Low	Nominal

Software Equation

- Software equation is a **dynamic multivariable model** that assumes a specific distribution of effort over the life of **Software Development Project**.

- The model has been derived from **productivity data** collected for over **4000 contemporary software projects**.

$$E = \frac{LOC \times B^{0.333}}{P^3} \times \frac{1}{t^4}$$

where

E = effort in person-months or person-years

t = project duration in months or years

B = "special skills factor"[13]

P = "productivity parameter" that reflects: overall proces

agement practices, the extent to which good software

TOPIC – 5
Estimation for Object Oriented Projects

Estimation for Object Oriented Projects

- To supplement **conventional software cost estimation methods with a technique that has been designed explicitly for OO software**.
- **Develop estimates** using effort decomposition, FP analysis, and any other method that is applicable for conventional applications.
- Using the requirements model, **develop use cases** and **determine a count**.
- From the requirements model, **determine the number of key classes**.
- **Categorize the type of interface for the application** and develop a multiplier for support classes.

- **Multiply the total number of classes (key + support) by the average number of work units per class**. Lorenz and Kidd suggest 15 to 20 person-days per class.

- **Cross-check the class-based estimate** by **multiplying the average number of work units per use case**.

Solution

Functional Units	Count	Complexity		Complexity Totals
External Inputs (EIs)	10	Low x 3	=	30
	15	Average x 4	=	60
	17	High x 6	=	102
External Outputs (EOs)	6	Low x 4	=	24
	0	Average x 5	=	0
	13	High x 7	=	91
External Inquiries (EQs)	3	Low x 3	=	9
	4	Average x 4	=	16
	2	High x 6	=	12
External logical Files (ILFs)	0	Low x 7	=	0
	2	Average x 10	=	20
	1	High x 15	=	15
External	5		=	45

- Adjustment factor

$$\sum_{i=1}^{14} F_i = 3+4+3+5+3+3+3+3+3+3+2+3$$

$$CAF = (0.65 + 0.01 \times \Sigma F_i)$$
$$= (0.65 + 0.01 \times 41)$$

$$= 424 \times 1.06$$
$$= 449.44$$

Hence $FP = 4$

Problem-Based Estimation

1) Start with a bounded statement of scope
2) Decompose the software into problem functions that can
   each be estimated individually
3) Compute an LOC or FP value for each function
4) Derive cost or effort estimates by applying the LOC or FP values to your baseline productivity metrics (e.g., LOC/person-month or FP/person-month)
5) Combine function estimates to produce an overall estimate for the entire project

(More on next slide)

Problem-Based Estimation (continued)

- In general, the LOC/pm and FP/pm metrics should be computed by project domain
  - Important factors are team size, application area, and complexity
- LOC and FP estimation differ in the level of detail required for decomposition with each value
  - For LOC, decomposition of functions is essential and should go into considerable detail (the more detail, the more accurate the estimate)
  - For FP, decomposition occurs for the five information domain characteristics and the 14 adjustment factors
    - External inputs, external outputs, external inquiries, internal logical files, external interface files

pm = person month

Problem-Based Estimation (co

- For both approaches, the planner uses lessons learned to estimate an optimistic, most likely, and pessimistic size value for each function or count (for each information domain value)
- Then the expected size value S is computed as follows:

$$S = (S_{opt} + 4S_m + S_{pess})/6$$

- Historical LOC or FP data is then compared to S in order to cross-check it

Example

- The **best case** to complete this task is **120** man-hours (around 15 days). In this case, you have a talented team, they can finish the task in smallest time.
- The **most likely** case to complete this task is **170** man-hours (around 21 days). This is a normal case, you have enough resource and ability to complete the task
- The **worst case** to complete this task is **200** man-hours (around 25 days). You need to perform much more work because your team members are not experienced.

Example…

- Now, assign the value to each parameter as below

$$a = 120 \quad m = 170 \quad b =$$

- The effort to complete the task can be calculated using **double-triangular distribution** formula as follows-

$$E = (a + 4m + b)/6$$

$$E = (120 + 4 * 170 + 2($$

Example 2

- The organizational average productivity for systems of this type is 620 LOC/pm. Based on a burdened labor rate of $8000 per month, the cost per line of code is approximately $13. Based on the LOC estimate and the historical productivity data, Calculate the total estimated project cost.
- Ans:
- **Total Effort = Total LOC/Productivity = 33200/620=53.54 ≈ 54 person-months => 6 developers Effort = Total Effort/6 = 54/6 = 9 months => Total Cost = Total Effort * Labor Rate = 54 * 800 ≈ $43,200**

FP estimation-CAD software

Information domain value	Opt.	Likely	Pess.	Est. count	W
Number of external inputs	20	24	30	24	
Number of external outputs	12	15	22	16	
Number of external inquiries	16	22	28	22	
Number of internal logical files	4	4	5	4	
Number of external interface files	2	2	3	2	
Count total					

Process-Based Estimation

1) Identify the set of functions that the software needs to
   perform as obtained from the project scope
2) Identify the series of framework activities that need to be performed for each function
3) Estimate the effort (in person months) that will be required to accomplish each software process activity for each function

(More on next slide)

Process-Based Estimation (continued)

4) Apply average labor rates (i.e., cost/unit effort) to the effort estimated for each process activity
5) Compute the total cost and effort for each function and each framework activity
6) Compare the resulting values to those obtained by way of the LOC and FP estimates
   - If both sets of estimates agree, then your numbers are highly reliable
   - Otherwise, conduct further investigation and analysis concerning the function and activity breakdown

Example

Activity →	CC	Planning	Risk analysis	Engineering		Construction release		CE
Task →				Analysis	Design	Code	Test	
Function								
↓								
UICF				0.50	2.50	0.40	5.00	n/
2DGA				0.75	4.00	0.60	2.00	n/
3DGA				0.50	4.00	1.00	3.00	n/
CGDF				0.50	3.00	1.00	1.50	n/
DBM				0.50	3.00	0.75	1.50	n/
PCF				0.25	2.00	0.50	1.50	n/
DAM				0.50	2.00	0.50	2.00	n/
Totals	0.25	0.25	0.25	3.50	20.50	4.50	16.50	

Example…

- It should be noted that 53 percent of all effort is expended on front-end engineering tasks (requirements analysis and design).
- Based on an average burdened labor rate of $8,000 per month, the total estimated project cost is $368,000 and the estimated effort is 46 person-months..

Estimation with the Use Cases

Each person will have each use cases
The estimation is done through the below formula

$$\text{LOC estimate} = N \times \text{LOC}_{\text{avg}} + [(S_a/S_h - 1) + (P_a/P_h - 1)] \times \text{LOC}_{\text{adjust}} \qquad (26.2)$$

where

N	= actual number of use cases
LOC_{avg}	= historical average LOC per use case for this type of subsystem
$\text{LOC}_{\text{adjust}}$	= represents an adjustment based on n percent of LOC_{avg} where n is defined locally and represents the difference between this project and "average" projects
S_a	= actual scenarios per use case
S_h	= average scenarios per use case for this type of subsystem
P_a	= actual pages per use case
P_h	= average pages per use case for this type of subsystem

Example

	use cases	scenarios	pages	scenarios	pages	LOC	LOC estimate
User interface subsystem	6	10	6	12	5	560	3,366
Engineering subsystem group	10	20	8	16	8	3100	31,233
Infrastructure subsystem group	5	6	5	10	6	1650	7,970
Total LOC estimate							42,568

n = 30 % , LOC average = 560
N= number of use cases for 1st sub system is 6
LOC avg = 560 , sa=10, sh=12, pa=6, ph =5
loc adjust = 30% of 560
apply all these in the formula stated in the slide 26

LOC estimate = 6*(560)+[(10/12 -1)+(6/5-1]*(30% of 560)
= 6*560 +(-0.16+0.2)*240
= 3360+0.04*240
=3367.2 =approximately = to 3366.

Using 620 LOC/pm as the average productivity for systems of this type and a burdened labor rate of $8000 per month, the cost per line of code is approximately $13. Based on the use-case estimate and the historical productivity data, **the total estimated project cost is $552,000 and the estimated effort is 68 person-months.**

Estimation with use cases…

- Use cases are described using many different formats and styles and represent an external view (the user's view) of the software.
- Therefore, they can be written at many different levels of abstraction. Use cases do not address the complexity of the functions and features that are described,
- they can describe complex behavior (e.g., interactions) that involve many functions and features.

Steps to compute use case estimation

- To compute *use case points (UCPs) –similar to function point*
- Computation of use case points must take the following characteristics

✓ The number and complexity of the use cases in the system.

✓The number and complexity of the actors on the system.

✓Various nonfunctional requirements (such as portability, performance, maintainability) that are not written as use cases.

✓The environment in which the project will be developed (e.g., the programming language, the software team's motivation).

Complexity factors for each usecase

Simple	**Simple UI**	**Single DB**	**<=3 transactions**	**<5 classes**	**5**
Average	More complex UI	2 or 3 DB	4 to 7 transactions	5 to 10 classes	10
Complex	Complex UI	Multiple DB	>8 transactions	>11 classes	15

total *unadjusted use case weight (UUCW) is the sum* of all weighted counts

each actor is assessed as..

Simple actor	**another system, a machine or device, communicate through an API**	**1**
Average actors	automatons that communicate through a protocol or a data store	2
complex actors	humans who communicate through a GUI or other human interface	3

total *unadjusted actor weight (UAW) is* the sum of all weighted counts.

TCF and ECF..

- These unadjusted values are modified by considering technical complexity factors (TCFs) and environment complexity factors (ECFs). Thirteen factors contribute to an assessment of the final TCF, and eight factors contribute to the computation of the final ECF
- Final UCP value is computed in the following manner:

$$UCP = (UUCW + UAW) \times TCF \times ECI$$

The CAD softw tion 33.6.3 is composed of three groups: user interface subsystem (includes UICF), engineering subsy (includes the 2DGA, 3DGA, and DAM subsystems), and infrastructure group (includes CGDF and PCF subsystems). Sixteen complex use cas the user interface subsystem. The engineering subsystem group is d 14 average use cases and 8 simple use cases. And the infrastructure s described with 10 simple use cases. Therefore,

$$\text{UUCW} = (16 \text{ use cases} \times 15) + [(14 \text{ use cases} \times 10) + (8 \text{ use case} + (10 \text{ use cases} \times 5) = 470$$

Analysis of the use cases indicates that there are 8 simple actors, actors, and 4 complex actors. Therefore,

$$\text{UAW} = (8 \text{ actors} \times 1) + (12 \text{ actors} \times 2) + 4 \text{ actors} \times 3) = 44$$

After evaluation of the technology and the environment,

$$\text{TCF} = 1.04$$

$$\text{ECF} = 0.96$$

Using relationship 33.2,

$$\text{UCP} = (470 + 44) \times 1.04 \times 0.96 = 513$$

Using past project data as a guide, the development group has produ per UCP. Therefore, an estimate of the overall size of the CAD proj

Reconciling Estimates

- The results gathered from the various estimation techniques must be reconciled to produce a single estimate of effort, project duration, and cost
- If widely divergent estimates occur, investigate the following causes
  - The scope of the project is not adequately understood or has been misinterpreted by the planner
  - Productivity data used for problem-based estimation techniques is inappropriate for the application, obsolete (i.e., outdated for the current organization), or has been misapplied then reconcile the estimates
- The planner must determine the cause of divergence

Example

- The total estimated effort for the CAD software ranges from a low of 46 person-months (derived using a process-based estimation approach) to a high of 68 person-months (derived with use-case estimation)
- The average estimate (using all four approaches) is 56 person-months.
- The variation from the average estimate is approximately 18 percent on the low side and 21 percent on the high side

Empirical Estimation Models

- Estimation models for computer software use empirically derived formulas to predict effort as a function of LOC (line of code) or FP(function point)
- Resultant values computed for LOC or FP are entered into an estimation model
- The empirical data for these models are derived from a limited sample of projects
  - Consequently, the models should be calibrated to reflect local software development conditions

$$E = A + B \times (e_v)^C \qquad (26.3)$$

where A, B, and C are empirically derived constants, E is effort in person-months, and e_v is the estimation variable (either LOC or FP

many LOC-oriented estimation models proposed in the literature are

$E = 5.2 \times (\text{KLOC})^{0.91}$	Walston-Felix model
$E = 5.5 + 0.73 \times (\text{KLOC})^{1.16}$	Bailey-Basili model
$E = 3.2 \times (\text{KLOC})^{1.05}$	Boehm simple model
$E = 5.288 \times (\text{KLOC})^{1.047}$	Doty model for KLOC > 9

FP-oriented models have also been proposed. These include

$E = -91.4 + 0.355\ \text{FP}$	Albrecht and Gaffney model
$E = -37 + 0.96\ \text{FP}$	Kemerer model
$E = -12.88 + 0.405\ \text{FP}$	Small project regression model

COCOMO

- Stands for Constructive Cost Model
- Introduced by Barry Boehm in 1981 in his book "Software Engineering Economics"
- Became one of the well-known and widely-used estimation models in the industry
- It has evolved into a more comprehensive estimation model called COCOMO II
- COCOMO II is actually a hierarchy of three estimation models
- As with all estimation models, it requires sizing information and accepts it in three forms: object points, function points, and lines of source code

(More on next slide)

COCOMO –II

- uses object points
- *object point* *is an indirect software measure that is computed* using counts of the number of (1) screens (at the user interface), (2) reports, and (3) components likely to be required to build the application.
- Each object instance (e.g., a screen or report) is classified into one of three complexity levels (i.e., simple, medium, or difficult) using criteria suggested by Boehm
- Once complexity is determined, the number of screens, reports, and components are weighted according to the table illustrated
- The object point count is then determined by multiplying the original number of object instances by the weighting factor and summing to obtain a total object point count

Number of views contained	*# and sources of data tables*		
	Total < 4 (< 2 server < 3 client)	*Total < 8 (2 – 3 server 3 – 5 client)*	*T (> >*
< 3	Simple	Simple	M
3 – 7	Simple	Medium	I
> 8	Medium	Difficult	I

Number of sections contained	*# and sources of data tables*		
	Total < 4 (< 2 server < 3 client)	*Total < 8 (2 – 3 server 3 – 5 client)*	*To (> 3 > 5*
0 or 1	Simple	Simple	M
2 or 3	Simple	Medium	D
4 +	Medium	Difficult	D

Object Type	*Complexity Weight*		
	Simple	*Medium*	
Screen	1	2	
Report	2	5	
3GL Component	—	—	

COCOMO -II

When the software is reused ...then percentage of reuse is estimated and the object point count is adjusted by using the formula

- NOP = (object points) X [(100 - %reuse)/100]

where NOP is defined as new object points.

- To derive an estimate of effort based on the computed NOP value, a "productivity rate" must be derived.

$$\text{PROD} = \frac{\text{NOP}}{\text{person-month}}$$

Developers experience & capability	Productivity (PROD)
Very Low	4
Low	7
Nominal	13
High	25
High	50

Productivity Rate

- Once the productivity rate has been determined, an estimate of project effort.

$$\text{Estimated effort} = \frac{\text{NOP}}{\text{PROD}}$$

- **Step-1:**
  Number of screens = 4
  Number of records = 2
- **Step-2:**
  For screens,
  Number of views = 4
  Number of data tables = 7
  Number of servers = 3
  Number of clients = 4
- by using above given information and table (For Screens),
  Complexity level for each screen = medium

Number of sections = 6
Number of data tables = 7
Number of servers = 2
Number of clients = 3
by using above given information and table (For Reports),
Complexity level for each report = difficult

Step-3:

By using complexity weight table we can assign complexity weight to each object instance depending upon their complexity level.
Complexity weight for each screen = 2
Complexity weight for each report = 8

- **Step-4:**
- Object point count = sigma (Number of object instances) * (its Complexity weight) = 4 * 2 + 2 * 8 = 24
- **Step-5:**
- %reuse of object points = 10% (given)
- NOP = [object points * (100 - %reuse)]/100
- = [24 * (100 -10)]/100 = 21.6
- **Step-6:**

  Developer's experience and capability is low (given)

  Using information given about developer and productivity rate table

  Productivity rate (PROD) of given project = 7

Developers experience & capability	Productivity (PROD)
Very Low	4
Low	7
Nominal	13
High	25
High	50

Productivity Rate

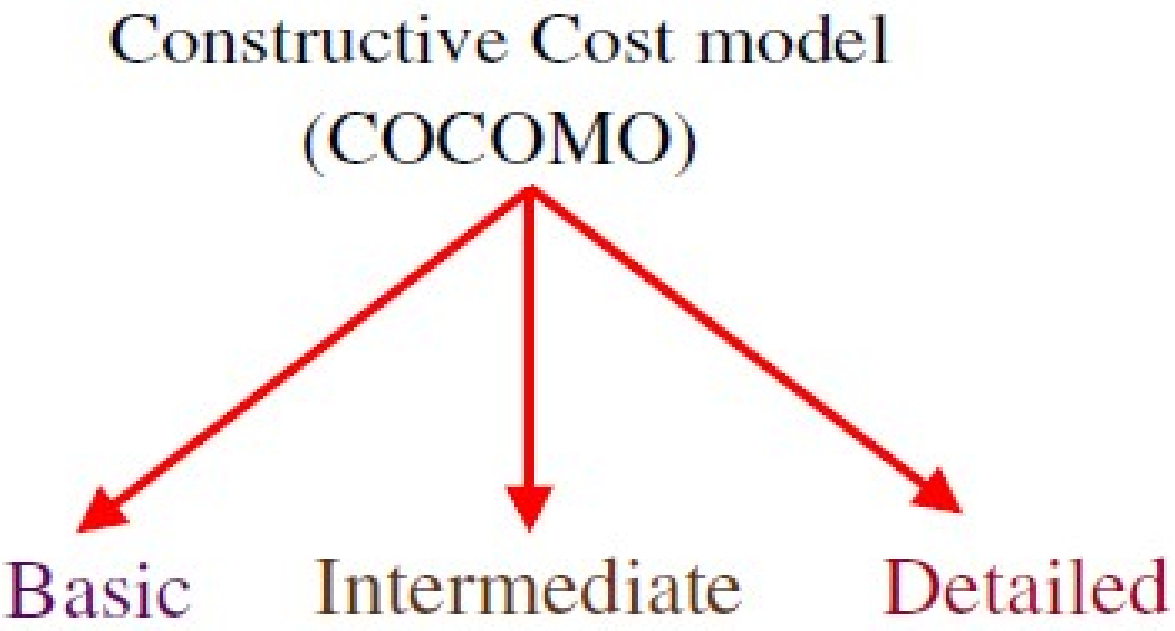
Step-7:

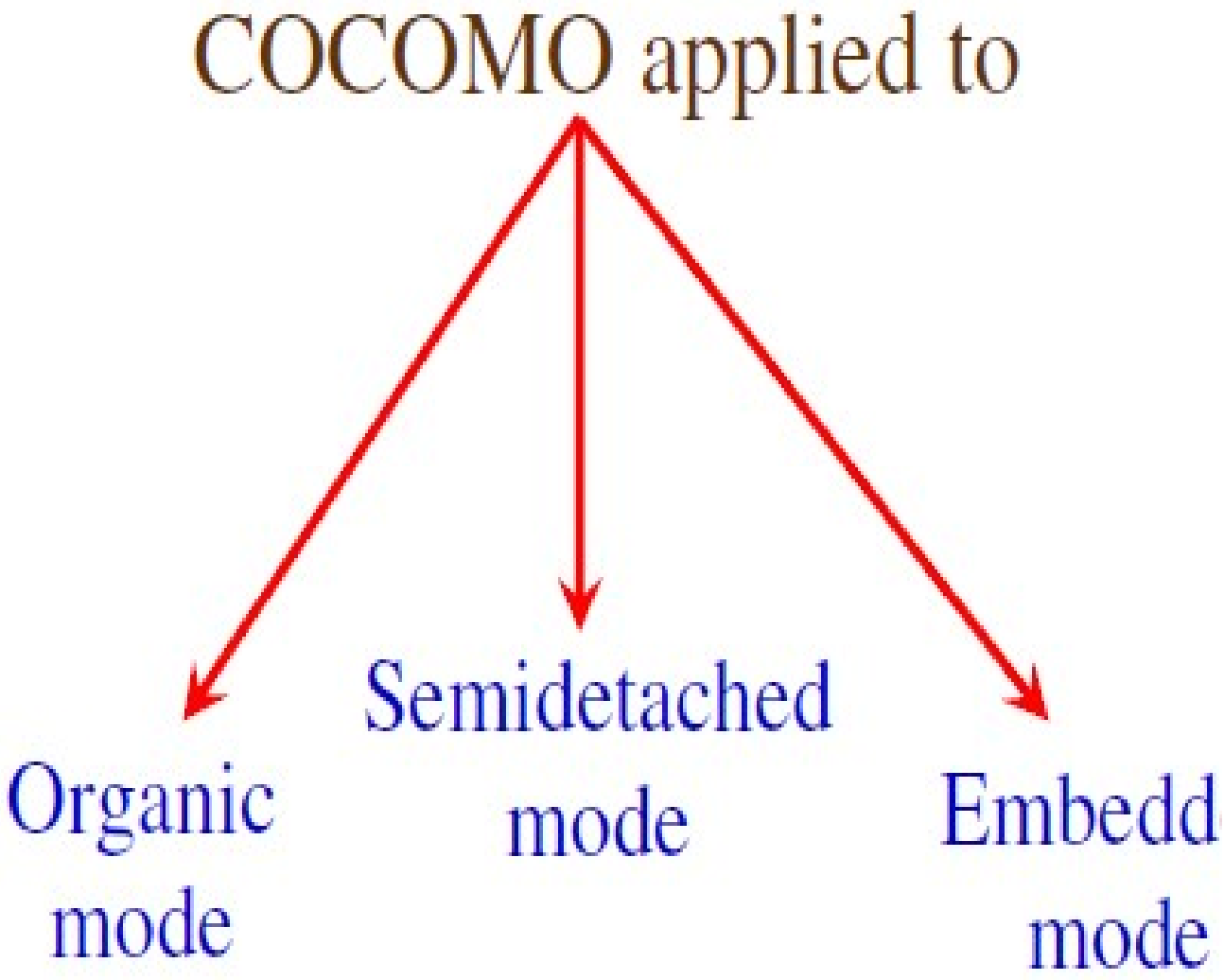
Effort = NOP/PROD = 21.6/7 = 3.086 person-month

Therefore, effort to develop the given project = 3.086 person-month.

COCOMO Models

The Constructive Cost Model (COCOMO)




COCOMO applied to
Organic mode
Semidetached mode
Embedd mode

Comparison of 3 COCOMO models

Mode	Project size	Nature of Project	Innovation	Deadline the proje
Organic	Typically 2-50 KLOC	Small size project, experienced developers in the familiar environment. For example, pay roll, inventory projects etc.	Little	Not tight
Semi detached	Typically 50-300 KLOC	Medium size project, Medium size team, Average previous experience on similar project. For example: Utility systems like compilers, database systems, editors etc.	Medium	Medium
Embedded	Typically over	Large project, Real time	Significant	Tight

Basic Model

Basic COCOMO model takes the form

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

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

Software Project	a_b	b_b	c_b
Organic	2.4	1.05	2.5
Semidetached	3.0	1.12	2.5

Example

Suppose that a project was estimated to be 4 Calculate the effort and development time for each modes i.e., organic, semidetached and embedded.

Solution

The basic COCOMO equation take the form:

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

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

Estimated size of the project = 400 KLOC

(i) Organic mode

$$E = 2.4(400)^{1.05} = 1295.31 \text{ PM}$$

(ii) Semidetached mode

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

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

(iii) Embedded mode

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

Example 2 (HW)

- A project size of 200 KLOC is to be developed. Software development team has average experience on similar type of projects. The project schedule is not very tight. Calculate the effort, development time.

Solution

The semi-detached mode is the most appropriate mode; view the size, schedule and experience of the developm

Hence $E = 3.0(200)^{1.12} = 1133.12$ PM

$D = 2.5(1133.12)^{0.35} = 29.3$ PM

Q.No.1: **Assume that you have been contacted by an Organization t Software, after applying the size-oriented metrics an estimate of 33200 L Using COCOMO model develop an effort, duration and no-of-person f Consider the**

$a_b = 2.4$, $b_b = 1.05$, $c_b = 2.5$, $d_b = 0.35$

Given:

LOC = 33200 = 33.2 k

FOR CAD SOFTWARE :

($a_b = 2.4$, $b_b = 1.05$, $c_b = 2.5$, $d_b = 0.35$)

FORMULA:

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

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

No. of person = E/D

SOLUTION:

The Software Equation

- The *software equation [Put92] is a dynamic multivariable model that assumes a* specific distribution of effort over the life of a software development project.
- Derived from productivity data collected for over 4,000 contemporary software projects

The Software Equation

$$E = [LOC \times B^{0.333}/P]^3 \times (1/$$

where

E = effort in person-months or per

t = project duration in months or y

B = "special skills factor"

B increases slowly as "the need for integration, testing, quality assurance, docun management skills grows" [Put92]. For small programs (KLOC = 5 to 15), *B* = 0.16. greater than 70 KLOC, *B* = 0.39.

Productivity parameter

- Reflects:

✓overall process maturity and management practices,

✓the extent to which good software engineering practices are used,

✓the level of programming languages used,

✓the state of the software environment,

✓the skills and experience of the software team,

✓and the complexity of the application

Typical value of P

- ✓*P = 2,000 for development of real-time embedded software,*
- ✓*P = 10,000 for telecommunication and systems software,*
- ✓P=12000 for scientific software (CAD)
- ✓ *P=28,000* for business systems applications
- The productivity parameter can be derived for local conditions using historical data collected from past development efforts.

Software equation…

The software equation has two independent parameters:

- (1) an estimate of size (in LOC) and
- (2) an indication of project duration in calendar months or years.

To simplify the estimation process, Putnam and Myers [Put92] suggest a set of equations derived from the software equation.

Minimum development time is defined as

$$t_{min} = 8.14\left(\frac{LOC}{p}\right)^{0.43} \quad \text{in months for } t_{\min} > 6 \text{ months}$$

$$E = 180\,Bt^3 \text{ in person-months for } E \geq 20 \text{ persor}$$

Where t is represented in years

Example

- Consider the CAD software
- Recommended value of P=12000 (for scientific software)

$$t_{min} = 8.14\left(\frac{33200}{12000}\right)^{0.43} = 12.6 \text{ calendar}$$

$$E = 180 \times 0.28 \times (1.05)^3 = 58 \text{ person-mo}$$

Example (HW)

b): You have been asked to develop a business system a

months time. Using software equation calculates the effort requir

Consider the following parameters for the system developme

- **for small program KLOC = 5 to 15 .....**
- **for program greater than = 70 KLOC .....**
- **for embedded software $P = 2000$**
- **for Telecomm and system software $P = 10000$**
- **for Business application $P = 8000$**

Given:

LOC = 15000

B = 0.16

P = embedded software = 2000

$$t_{min} = 8.14 \times \left(\frac{5600}{6000}\right)^{0.43}$$

$$= 10.67 \text{ months}$$

$$\simeq 11 \text{ months}$$

$$E = 18 \times 0.16 \times (0.5)^3$$

$$= 36 \text{ persons}$$

$$\simeq 4 \text{ "}$$

ESTIMATION FOR OBJECT-ORIENTED PROJECTS

Lorenz and Kidd [Lor94] suggest the following approach:

1. Develop estimates using effort decomposition, FP analysis, and any other method that is applicable for conventional applications.
2. Using the requirements model, develop use cases and determine a count. Recognize that the number of use cases may change as the project progresses.

Estimation of OO projects…

3. From the requirements model, determine the number of key classes.(Analysis classes).
4. Categorize the type of interface for the application and develop a multiplier for support classes

Interface type	M
No GUI	
Text-based user interface	
GUI	
Complex GUI	

Estimation of OO projects

5. Multiply the number of key classes (step 3) by the multiplier to obtain an estimate for the number of support classes.
6. Multiply the total number of classes (key + support) by the average number of work-units per class.

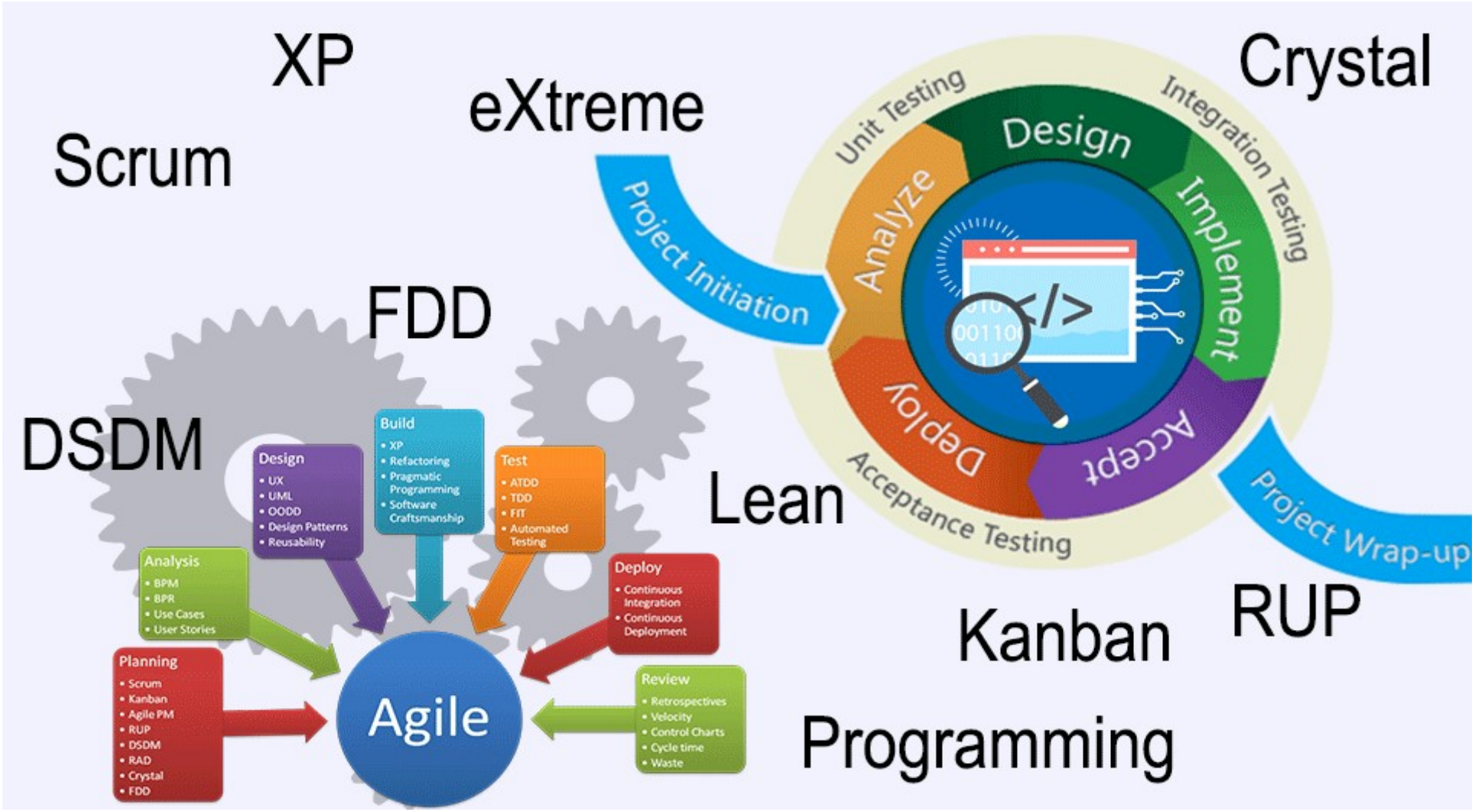
Lorenz and Kidd suggest 15 to 20 person-days per class

- Cross check the class-based estimate by multiplying the average number of work-units per use-case!

SAS
ENGINEERING · MANAGEM
DEEMED T
(U/S 3 OF

Thank you

Agile models



Scrum

Scrum as a better way of building products.

Scrum is a lightweight framework, simple to understand but difficult to master.

Scrum is used everywhere. (90% of agile team use scrum, it has only on scrum guide)

Scrum incorporates a set of process patterns that emphasiz project priorities, compartmentalized work units, communicatio and frequent customer feedback.

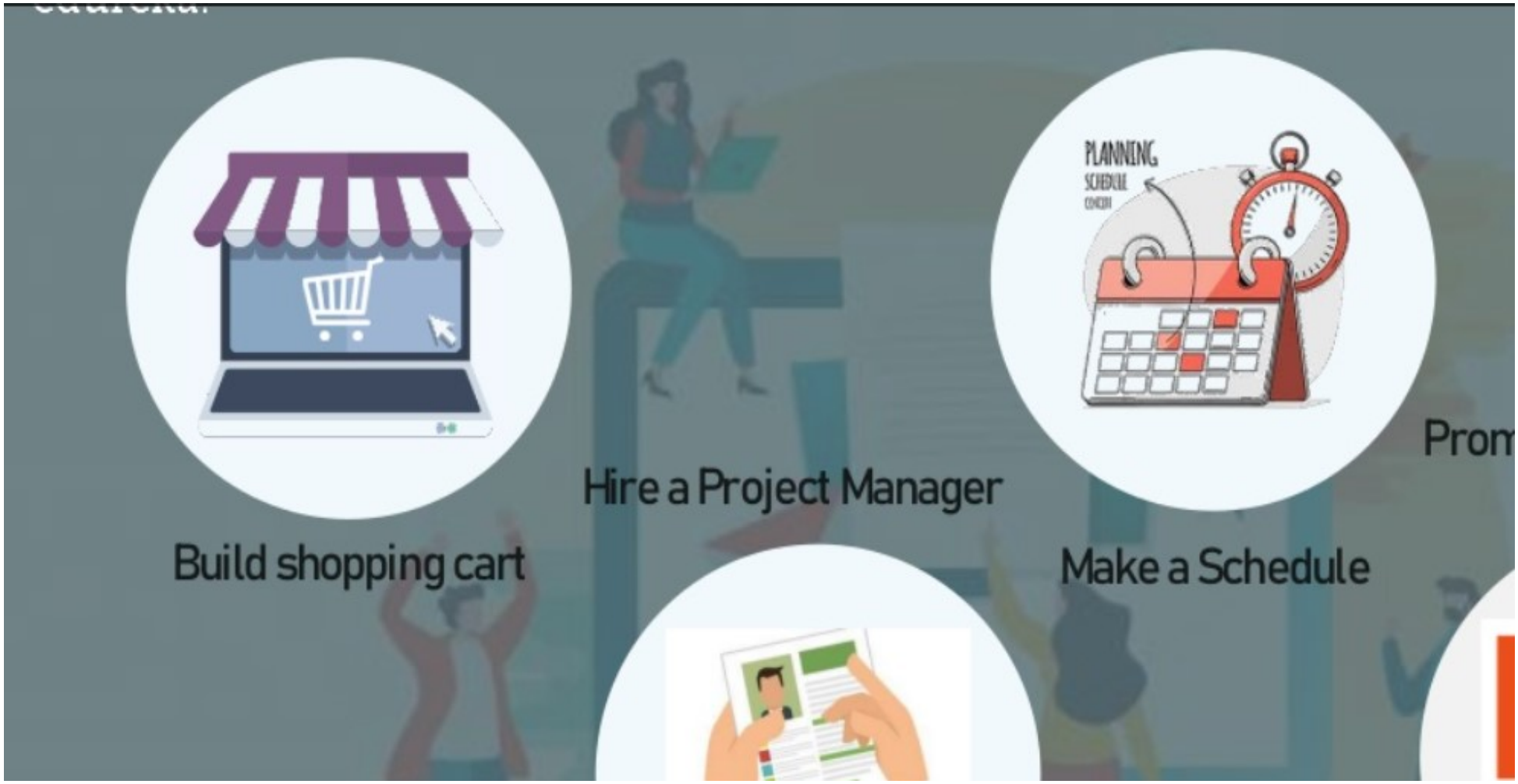
Scrum


"Scrum is an agile framework within which people can addr
problems, while productively and creatively delivering product
value."

Scrum

- Meetings are very short (15 minutes daily) and sometimes conducted witho chairs (what did you do since last meeting? What obstacles are yo encountering? What do you plan to accomplish by next meeting?) – scru board
- "demos" are delivered to the customer with the time-box allocated. May n contain all functionalities. So customers can evaluate and give feedbacks.
- 3 artifacts – product backlog, sprint backlog and increment (shippab product)
- 3 roles – Scrum team, scrum master, product owner
- Developer team is self organizing – cross functional team members

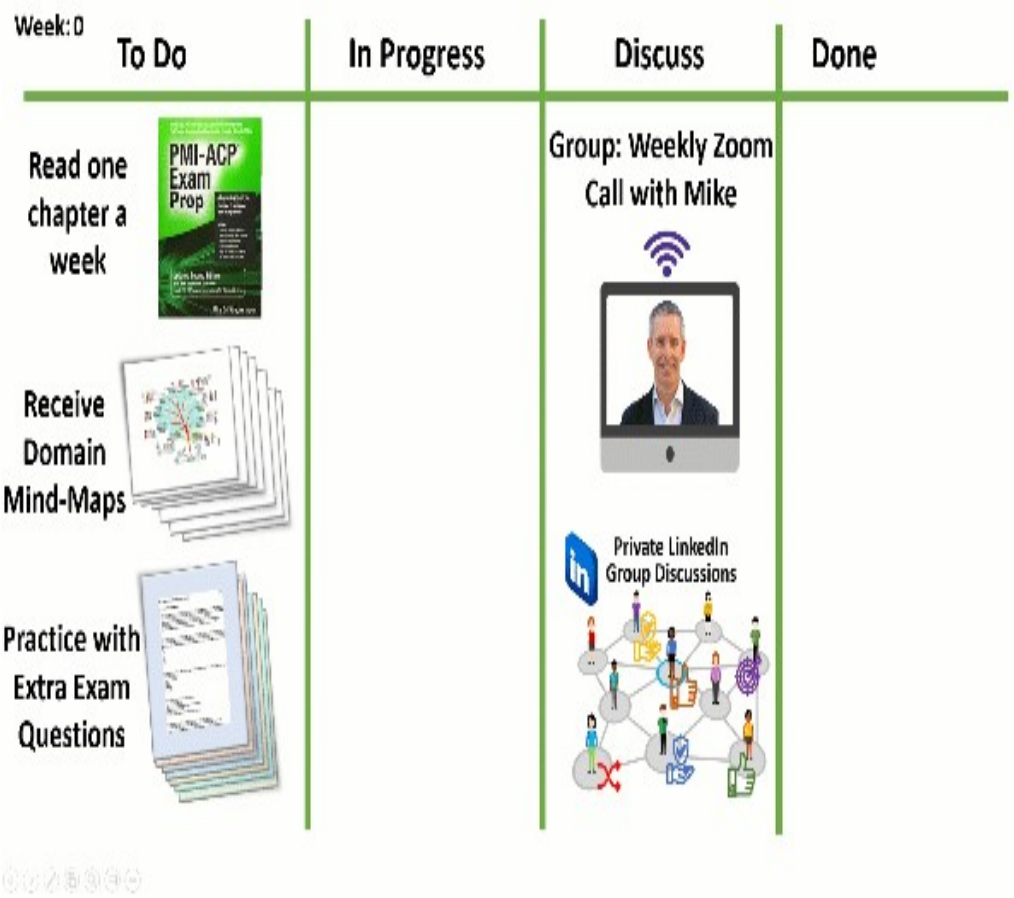
Scrum


Build shopping cart
Hire a Project Manager
PLANNING
Make a Schedule

Scrum Process

Scrum Board


The Scrum Board
Story
To Do
In Progress
To Verify


Week: 0
To Do
In Progress
Discuss
Done
Read one chapter a week
PMI-ACP Exam Prep
Group: Weekly Zoom Call with Mike
Receive Domain Mind-Maps
Private LinkedIn Group Discussions
Practice with Extra Exam Questions

LEAN

- Lean software development is a concept that emphasizes optimizi
  efficiency and minimizing waste in the software development proces
  This approach has its roots in the Lean manufacturing movement
  the 1980s. However, it is now considered an integral part of the Agi
  software development methodology.

Kanban

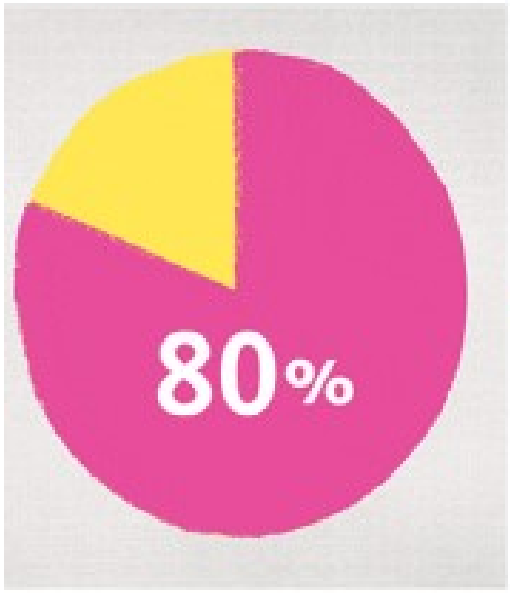
- Kanban is a popular framework used to implement agile and DevOps software development. It requires real-time communication of capacity and full transparency of work. Work items are represented visually on a kanban board, allowing team members to see the state of every piece of work at any time.

SCRUM	EXTREME PROGRAMMING (XP)
In Scrum framework, team work in iterations called Sprint which are 1-2 month long.	In Extreme Programming(XP), teamwork for 1-2 weeks only. Pair programming (2 members per team)
Scrum is the type of Agile framework. It is a framework within which people can address complex adaptive problem while productivity and creativity of delivering product is at highest possible values.	Extreme Programming is one of the most important models of Agile framework. This model emphasizes team-work and customer satisfaction as well.
Scrum model do not allow changes in their timeline or their guidelines.	Extreme Programming allow changes in their set timelines.
Scrum emphasizes self-organization.	Extreme Programming emphasizes strong engineering practices (pairprogramming,refactoring...)
In Scrum framework, team determines the sequence in which the product will be developed.	In Extreme Programming, team have to follow a strict priority order or pre-determined priority order.
Scrum framework is not fully described. If you want to adopt it then you need to fill the framework with your own frameworks method like XP, DSDM or Kanban.	Extreme Programming(XP) can be directly applied to a team. Extreme Programming is also known for its **Ready-to-apply** features.

https://www.visual-paradigm.com/scrum/extreme-programming-vs-scrum/

Dynamic Systems Development Method

4

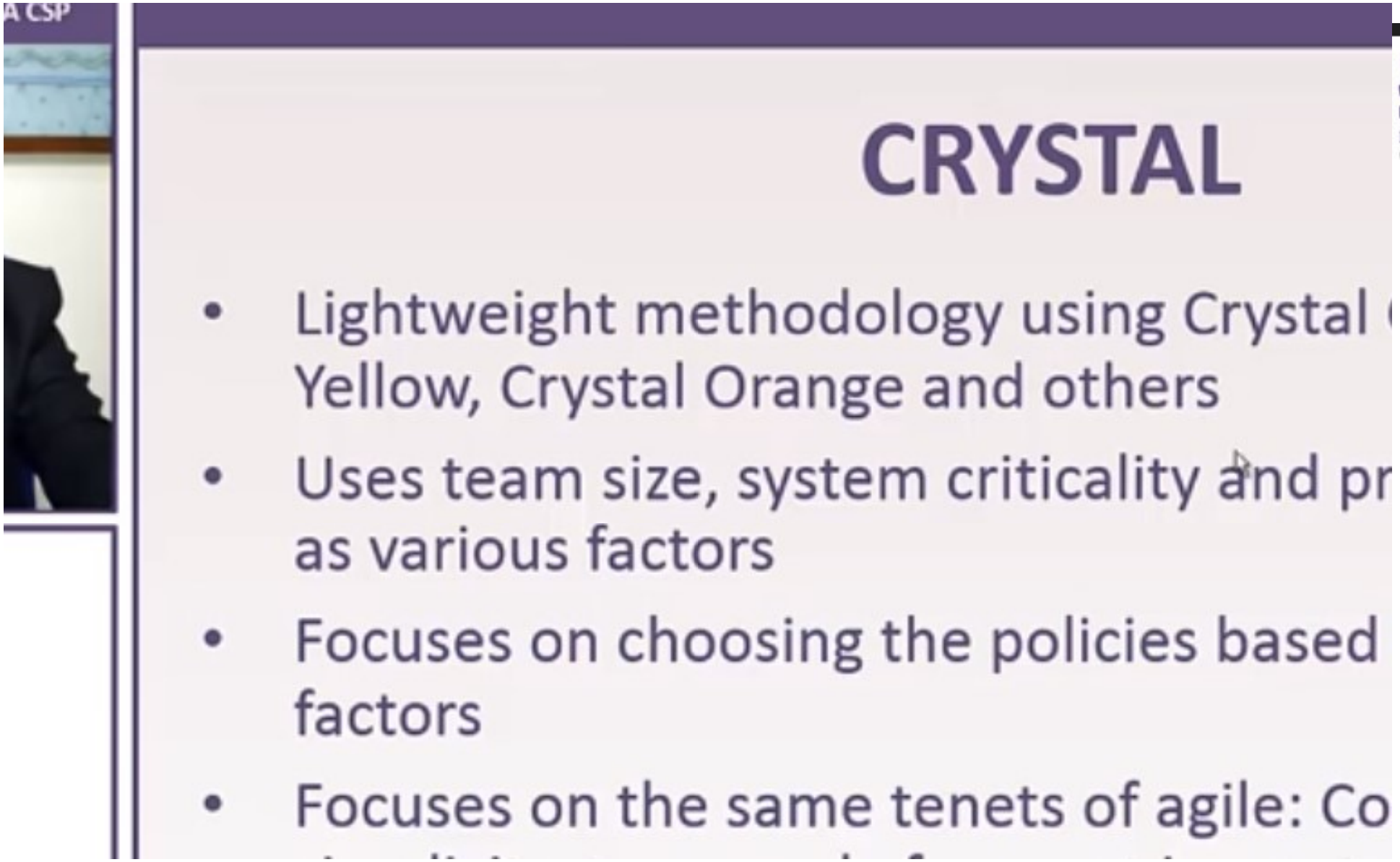


- A framework for building and maintai which meet tight time constraints throu incremental prototyping in a contr environment.
- The DSDM philosophy is borrowed fro version of the Pareto principle—80 p application can be delivered in 20 perce it would take to deliver the complete application.
- DSDM—distinguishing features
  - Similar in most respects to XP and/o

DSDM Principles

- Focuses on the entire lifecycle from Pre to Post Pro
  Implementation
- Primary focus on "Fitment of business purpose"
- Focuses on Principles below

Focus on Business Need	Deliver on Time
Collaborate (active user involvement)	Never compromise c
Build incrementally from firm foundations	Develop Iteratively



- L30 – means light criticality with 30 members
- Ex: L200 – hospital management s/w or NASA

Agile Modeling

- Available - a wide variety of software engineering modeling methods and notation have been proposed for analysis and design
- These methods have merit, but difficult to apply and challenging to sustain.
- Part of the problem is the "weight" of these modeling methods.
  - the volume of notation required,
  - the degree of formalism suggested,
  - the sheer size of the models for large projects,
  - the difficulty in maintaining the model(s) as

Agile Modeli

- Originally proposed by Scott Amt

- Definition:
- Agile Modeling (AM) is a practice-based methoc effective modeling and documentation of softwa systems.
- Agile Modeling (AM) is a collection of values, p and practices for modeling software that can be a a software development project in an effective weight manner.
- Agile models are more effective than traditiona because they are just barely good, they don't ha

Agile Modeling

Suggests a set of agile modeling principles

- Model with a purpose- Specific Goal
- Use multiple models- insight on small subs
- Travel light- long-term value
- Content is more important than representation- impart information (no need of flawed notations)

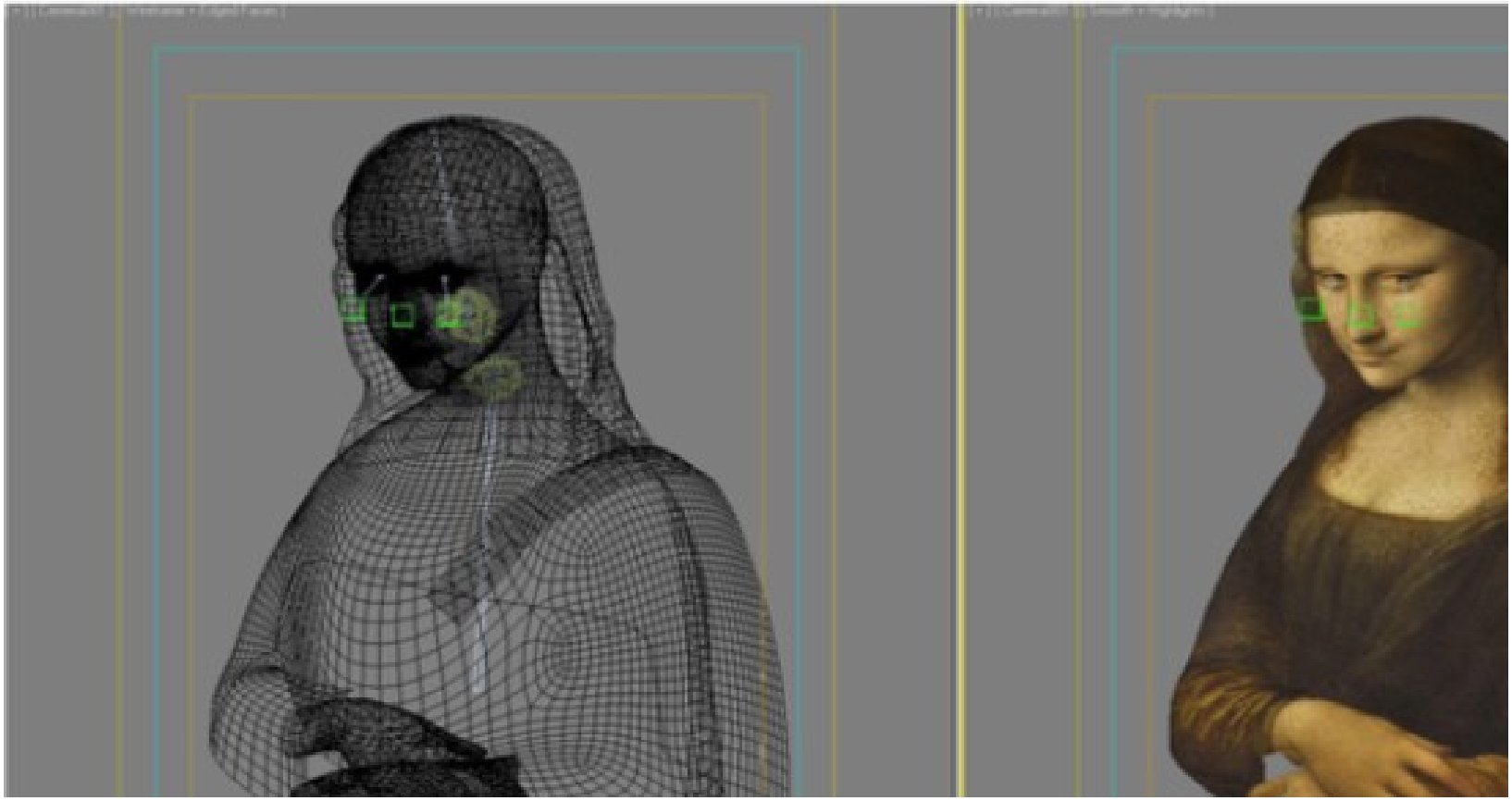
Agile Unified Process

38

- The *Agile Unified Process* (AUP) adopts a ' the large" and "iterative in the small" philos building computer-based systems.
- By adopting the classic UP phased a *inception, elaboration, construction,* and *trai*
- AUP provides a serial overlay (i.e., a sequence of software engineering activiti enables a team to visualize the overall proc for a software project.
- However, within each of the activities, th iterates to achieve agility and to deliver me software increments to end users as rapidly as possible.

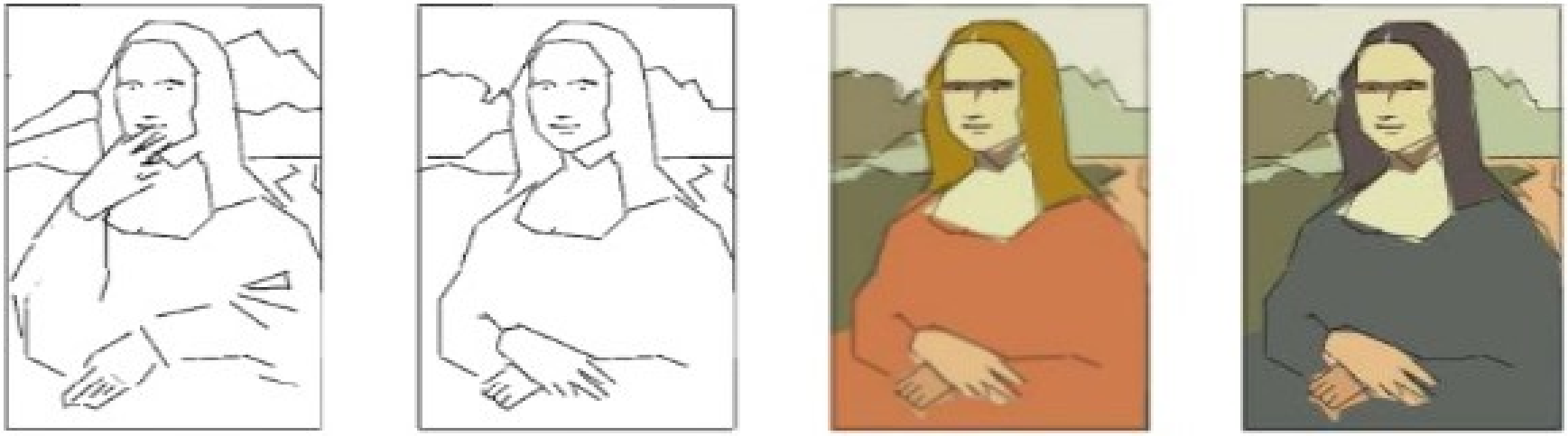
Waterfall

Prototyping Model

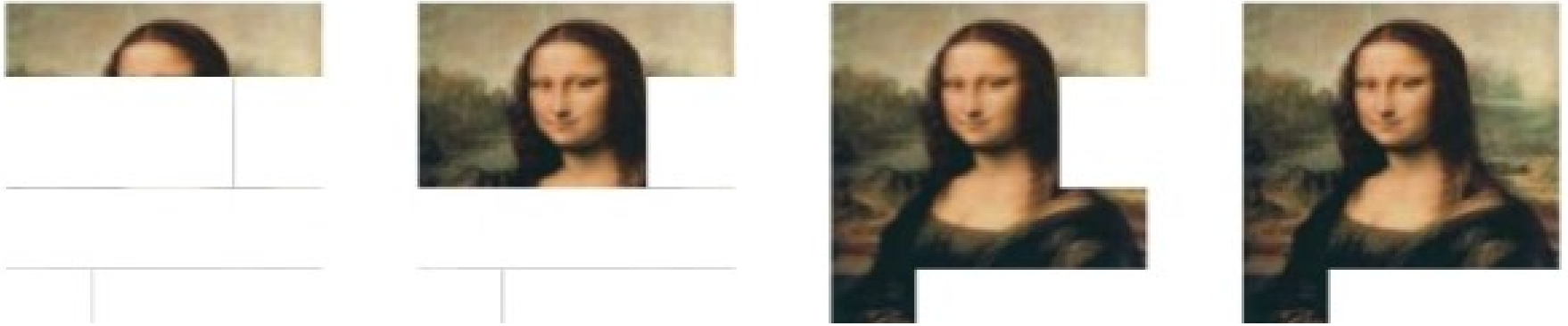


Iterative

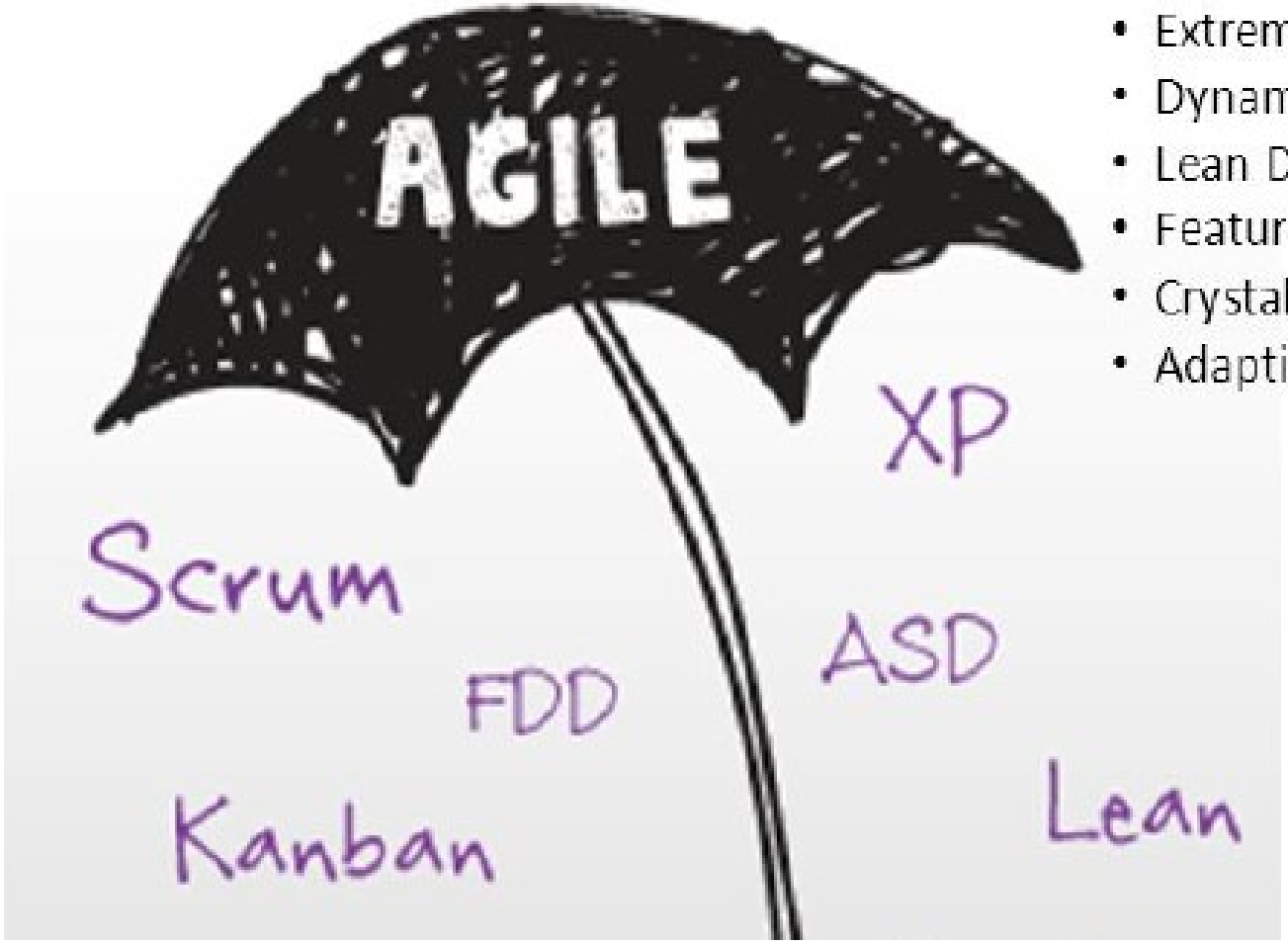
1 2 3 4



Incremental



Agile Methodologies



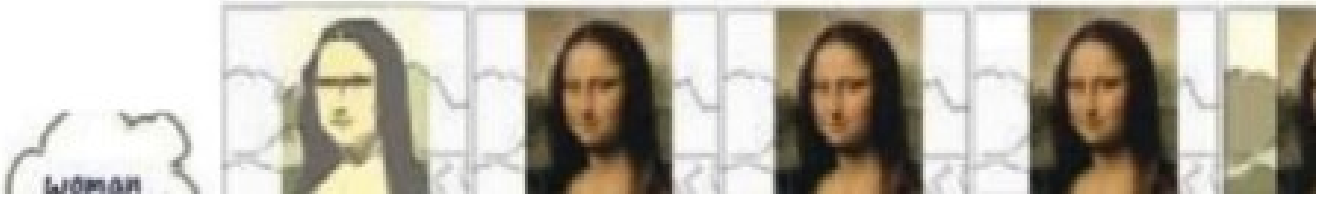
- Scrum
- Extreme Programming
- Dynamic System Development
- Lean Development
- Feature-Driven Development (
- Crystal
- Adaptive Software Developme

Methodology
Extreme Programming
Scrum
DSDM
Unified Process
FDD

Iterative

Incremental

Iterative & Incremental

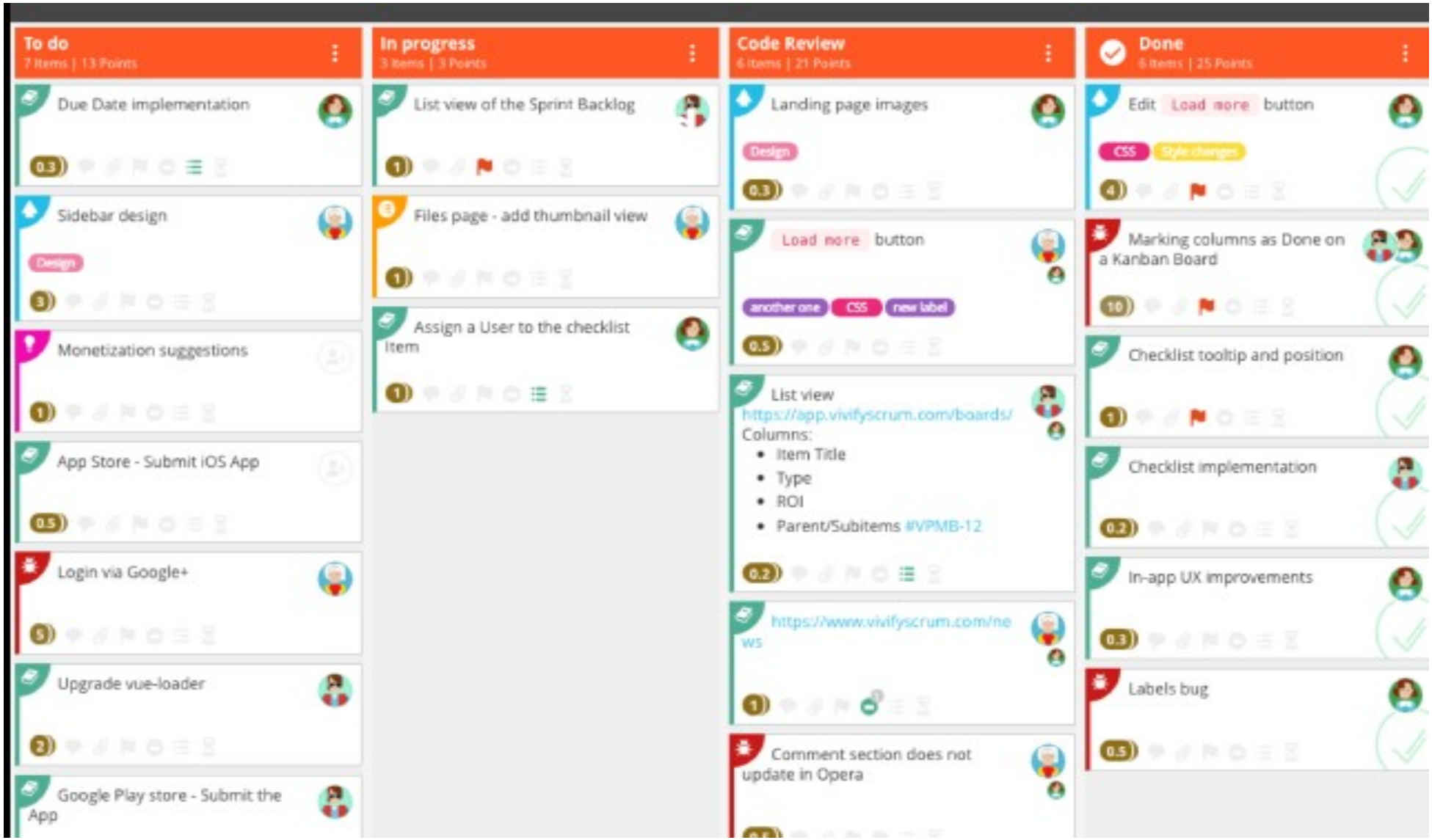


Iteration in Theory

Incrementalism in Theory

Agile as usually practiced

SCRUM BOARD


To do
7 Items | 13 Points
Due Date implementation
0.3
Sidebar design
Design
3
Monetization suggestions
1
App Store - Submit iOS App
0.5
Login via Google+
5
Upgrade vue-loader
2
Google Play store - Submit the App
In progress
3 Items | 3 Points
List view of the Sprint Backlog
1
Files page - add thumbnail view
1
Assign a User to the checklist item
1
Code Review
6 Items | 21 Points
Landing page images
Design
0.3
Load more button
another one
CSS
new label
0.5
List view
https://app.vivifyscrum.com/boards/
Columns:
Item Title
Type
ROI
Parent/Subitems #VPMB-12
0.2
https://www.vivifyscrum.com/news
1
Comment section does not update in Opera
Done
6 Items | 25 Points
Edit Load more button
CSS
Style changes
4
Marking columns as Done on a Kanban Board
10
Checklist tooltip and position
1
Checklist implementation
0.2
In-app UX improvements
0.3
Labels bug
0.5

Lean (Kanban)


BACKLOG
CURRENT SPRINT BACKLOG
DEVELOPMENT IN PROCESS
Feature Group A
Feature
Ready
In Process
3
Done
Story 1
Task 1.3
Task 1.2
Task 1.2
Feature Group B
Feature
Ready
In Process
3
Done
TICKET #2312
Defect 1
Task 2.3
Task 2.4
Task 2.2
FT
Task 2.7
Task 2.1
Task 2.5
Task 2.6
Feature Group C
Feature
Ready
In Process
3
Done
Story 3
Task 3.3
Task 3.4
FT
Task 3.2
Task 3.1

Scrum Vs Lean (Kanban)