#2 Software Size Estimation (Lecture-1)

The approaches → To measure size/value of the software (not size in MB/GB)

(i) Line of Code
- (*) Simply measure no. of Lines

(ii) Token Count (Halstead Metric)
- (*) Count operator and operand

→ Program Vocabulary (No. of unique operators + operands)

$$n = n1 + n2$$

→ Program Length (No. of total " " ")

$$N = N1 + N2$$

→ Program Value (Software size)

$$\boxed{V = N \log_2 n}$$

(iii) Function Point Analysis (FPA)
- Standardized methodology
- based on functionalities specified by the user.
- measured from user's pov
- measured in function points

→ Steps

→ (a) Identify counting boundary
- → border between application and external app with which my app communicates
- → function analysis done on my app only

→ (b) Identify data function and complexity
- Internal Logical File: Set of data within system
- External Interface File: Data to be sent to external apps

Ex: My data in a messaging app.

(c) Identify ~~external~~ transactional function complexity

(d) → Calculating UFP (unadjusted function point)

→ (Multiplying by ~~weight~~ and summing up no. of functions)

(e) → Find Value Adjustment Fact (VAF)

→ TDI = Weighted sum of 14 GSC (General system characteristic)

(f) Adjust function point, AFP = UFP X VAF (Ans)

→ I forgot, VAF = (TDI X 0.01) + 0.065

Note: For each function, you need to first know their type and complexity. Then sum up all the functions after multiplying by weight based on type and complexity (ILF, ELF etc.)

- Example-

Writing from 3rd party → [External Interface File | Average complexity]

Check the table to find score of this function

14 GSC are rated as no influence, incidental, moderate, average, significant, essential, for 0, 1, 2, 3, 4 and 5 respectively.

Comparism

	Waterfall / V-shaped	Iterative	Spiral	Prototype	Scrum
Requirements	• Clear • Doesn't change	• Mostly clear • Can change	• Unclear (Risky)	• Unclear • changes	
Delivery	Once	Many	Many	Once (but many prototypes)	Many
Documentation	Must (High)	Must (High)	Higher		Moderate
Planning	Detailed	High			Moderate
Team	Skilled	Unskilled	—		Unskilled
Tech	Fixed	New, changing	New	Checks feasibility	
Feedback	None	After Iteration	After prototypes	After prototype (early)	every 1-2 week
Disadvantages	→ Linear (cannot go back)	→ Early defn of complete → Well defined modules	→ High budget for smaller teams	→ Many prototypes (time + cost) → Unrealistic delivery dates after good prototype	
Advantages			→ Large projects with complex req → Risk assessment	→ complex req visualization → checks technical feasibility → early error + feature detection → save cost by removing unwanted features	