ETH Zürich
Chair of Software Engineering

Software Engineering

Algorithmic Estimation Techniques

Algorithmic estimation of software

- Basic cost model

$$\boxed{\text{Effort} = A \times \text{Size}^{B} \times m(X)}$$

Size: Some measurement of the software size
A: Constant factor that depends on
  Organizational practices
  Type of software
B: Usually lies between 1 and 1.5
X: Vector of cost factors
m: Adjustment multiplier

Cost models

$$\boxed{\text{Effort} = A \times \text{Size}^{B} \times m(X)}$$

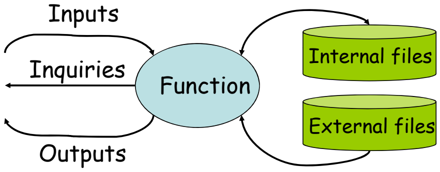
- Cost models
  - Define a way to determine the size
  - Define cost factors X
  - Provide defaults for parameters A, B, m (based on hundreds of projects)

Measuring size: Lines of code

- Software size can be measured in lines of source code
  - Most commonly used metric
- **Difficult in early phases** of the project (before design is known)
  - Reuse, make-or-buy decisions
- **Influenced** heavily by choice of **programming language**
- Should only be **used indirectly**

Function point analysis

- Size is estimated based on **requirements**



Functions

- **Inputs**
  - Forms, dialogs, messages, XML documents
- **Outputs**
  - Web pages, reports, graphs, messages, XML documents
- **Inquiries** (input/output combinations)
  - Simple web inputs, generally producing a single output
- **Logical internal files** (controlled by the program)
  - Tables, views or files in database
- **External files** (controlled by other programs)
  - Tables or files used from other systems or databases

Complexity of functions

Factor	Simple	Average	Complex
Inputs	3	4	6
Outputs	4	5	7
Inquiries	3	4	6
Ext. files	7	10	15
Int. files	5	7	10

- Determine **complexity** of each function
- **Weight** each function according to complexity

Input	Simple	Average	Complex
Data elements	1-5	6-10	>10
Checking	Formal	Formal, logical	Formal, logical, requires DB access

Cost factors

Data communications
Distributed processing
Performance
Heavy use
Transaction rate
Online data entry
Complex interface
Online data update
Complex processing
Reusability
Installation ease
Operational ease
Multiple sites
Facilitate change

Rate each element from 0 – 5
- 0: no influence
- 1: insignificant influence
- 2: moderate influence
- 3: average influence
- 4: significant influence
- 5: strong influence

Technical complexity factor
- TCF = 0.65 + 0.01 × sum
- Varies between 0.65 and 1.35

Function point computation

	Simple	Average	Complex
Inputs	6 x 3 = 18	2 x 4 = 8	3 x 6 = 18
Outputs	7 x 4 = 28	7 x 5 = 35	0 x 7 = 0
Inquiries	0 x 3 = 0	2 x 4 = 8	4 x 6 = 24
Ext. files	9 x 5 = 45	0 x 7 = 0	2 x 10 = 20
Int. files	5 x 7 = 35	2 x 10 = 20	3 x 15 = 45
Unadjusted function points (UFP)			304
Technical complexity factor (TCF)			1.15
Adjusted function points			**350**

Determining effort and size

- Empirical value for effort
  - Or use a table
- Empirical value for size
- Huge differences in productivity
  - Factor 10-20 between individual programmers
  - Factor 4 between companies

Effort = FP$^{1.4}$ / 150

Language	Level	Statements per UFP
Assembler	1	320
C	2.5	125
C++	6.5	50
Perl	15	25
Pascal	3.5	90
Visual Basic 3	10	30
Excel	50	6

Observation about software size

- Consider a project that requires 10 Web pages, 15 reports, and 20 database tables
  - 315 function points, if each item is medium complexity
- How many lines of C code would it have?
  - About 32,000 lines
- What if you used Excel?
  - About 2,000 lines

- Why do you think there are so many spreadsheets out there?

Function point analysis: Discussion

Pros
- Based on requirements (instead of code size)
- Can be applied in early project phases
- Can be calibrated (for company, project type)
- Counting standards by "International Function Points User Group"
- Technology-independent

Cons
- Estimation of overall effort (not per phase)
- Tailored towards functional decomposition (rather than OO)
- Tailored towards information systems
- Needs calibration to produce reliable results