Software Engineering
Effort Estimation

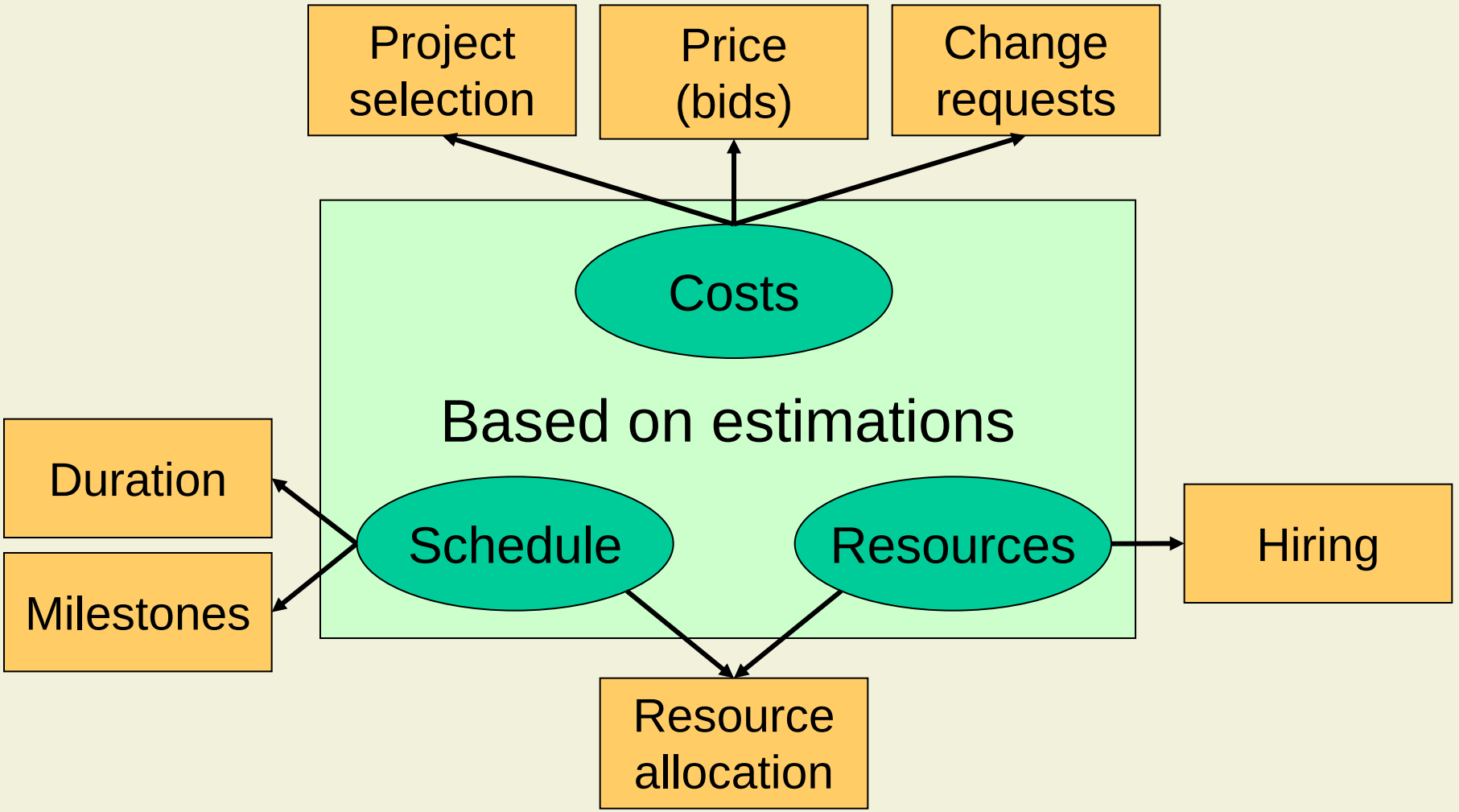
Prof. Dr. Peter Müller

Software Component Technology

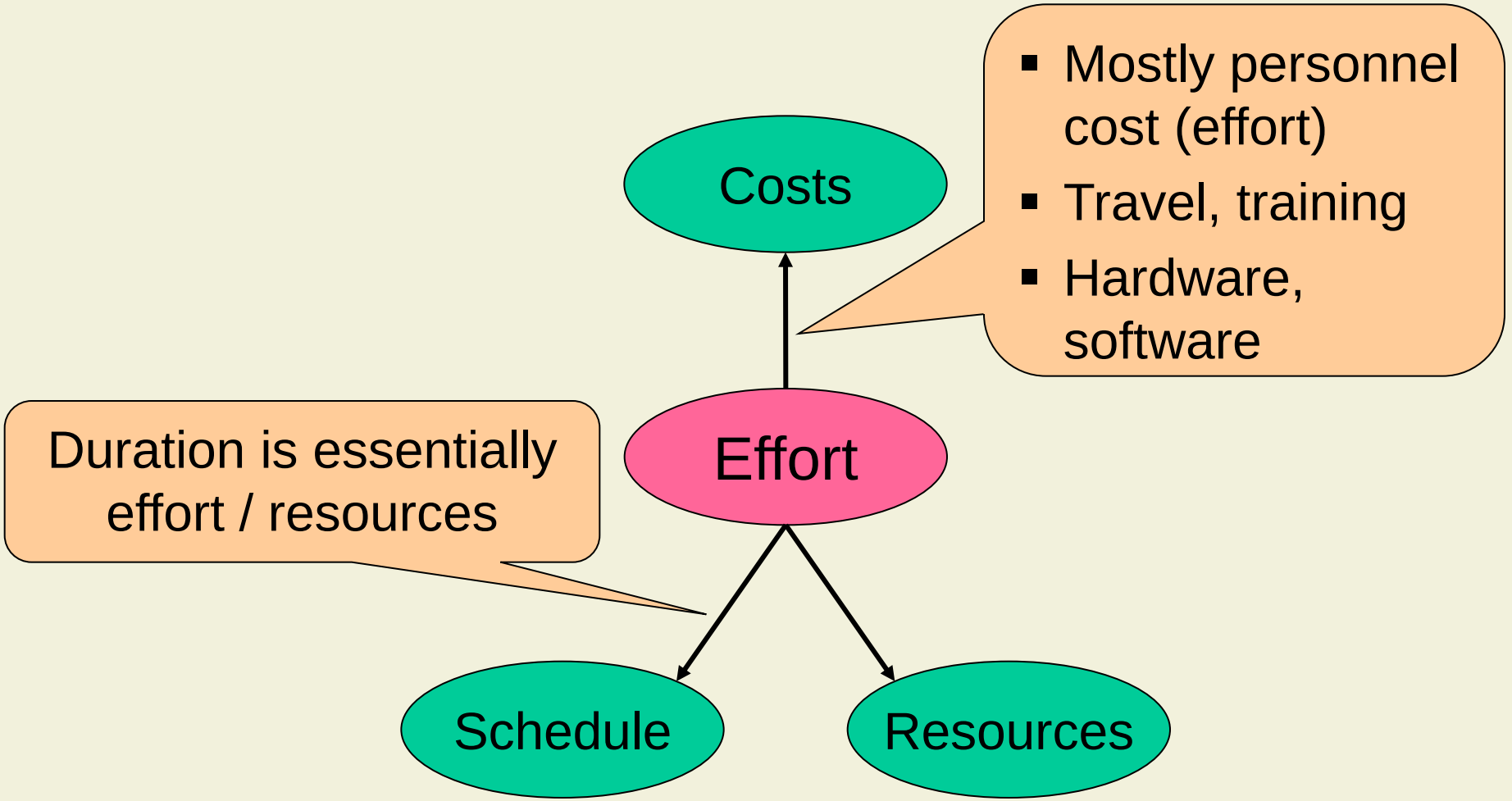
Summer Semester 06

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Why Estimations?



Estimations in Software Projects



9. Effort Estimation

Estimation Exercise

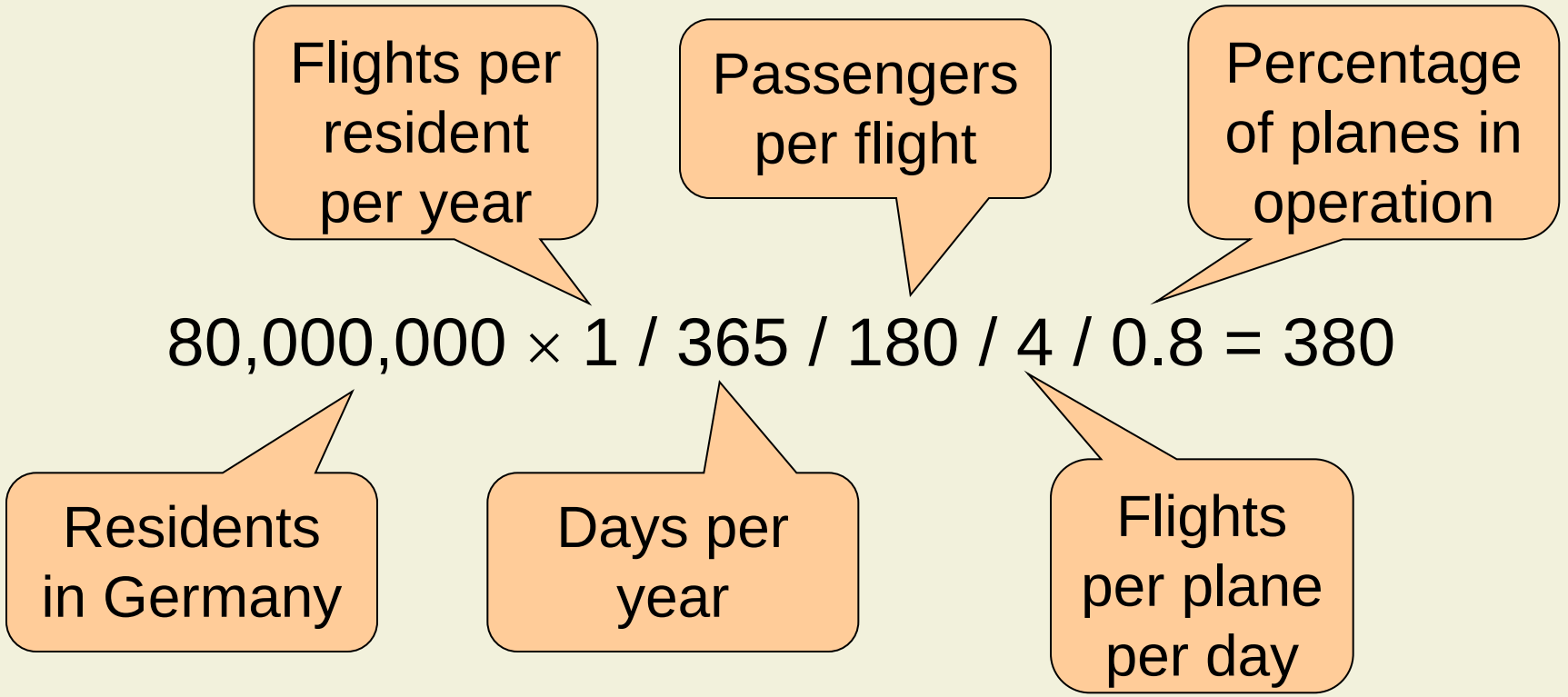
- How many passenger planes does Lufthansa have?
  - Not counting regional subsidiaries

- How can we approach this problem systematically?

Algorithmic Estimate 1

Exclude from pdf

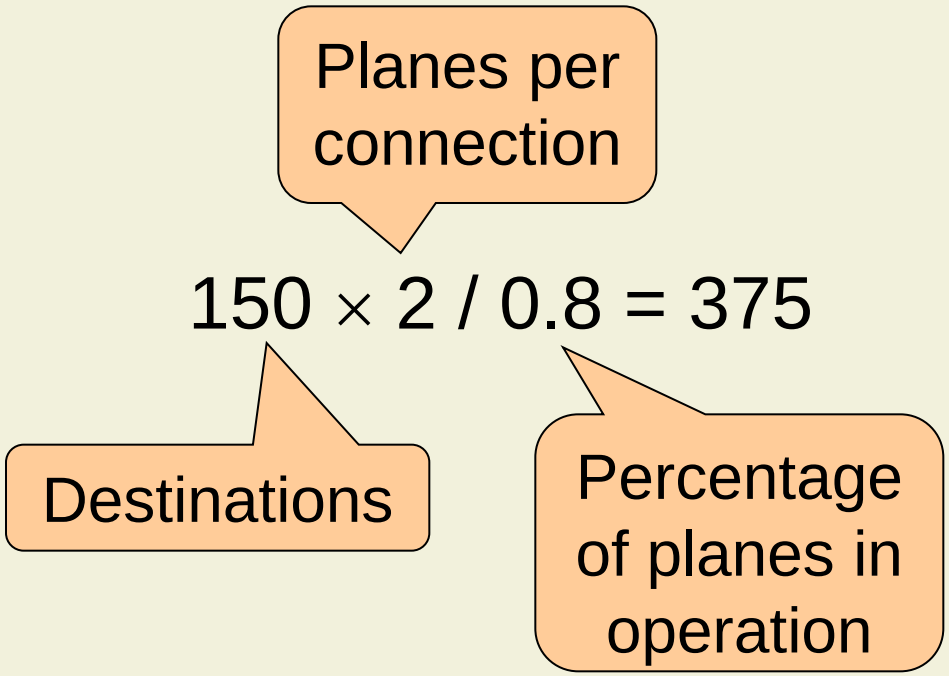
- Idea: Number of passengers corresponds to number of residents in the home country



Algorithmic Estimate 2

Exclude from pdf

- Idea: Number of planes corresponds to destinations
  - We assume a star topology



Estimation by Analogy

Exclude from pdf

- Idea: We know that Air France has 240 planes
  - We assume correlation with number of residents

Planes of Air France

$$240 / 60{,}000{,}000 \times 80{,}000{,}000 = 320$$

Residents in France

Residents in Germany

9. Effort Estimation

Empirical Estimation: Expert Judgment

- Estimate is based on **experience and historical data**
- Involve experts in
  - Development techniques
  - Application domain
- Most **common technique** in practice

Top-Down Estimation

- Estimation by **analogy**
  - Comparison with **similar projects**
  - Analysis of differences
  - Typical example: SAP introduction

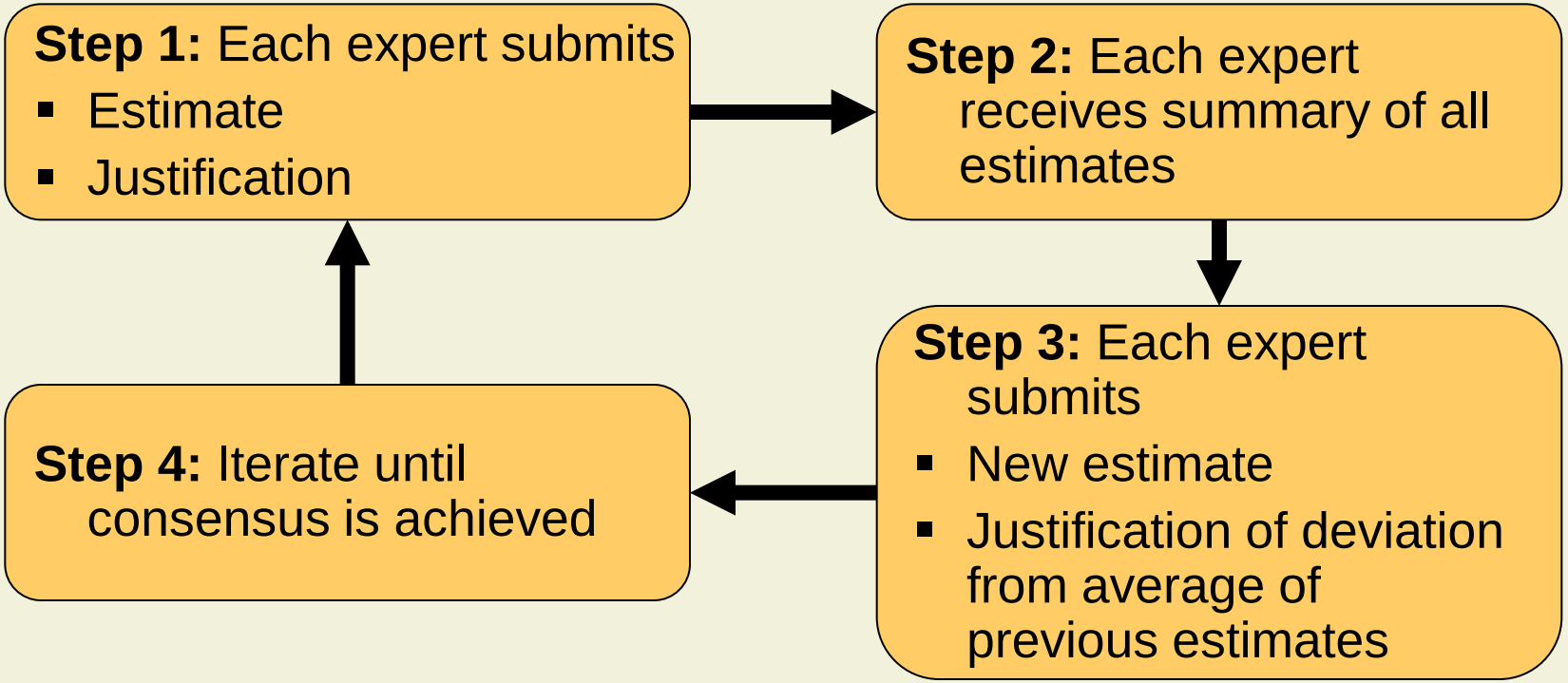
Pros

- Quicker and less expensive than other methods
- Can be done early in the project

Cons

- Underestimation of difficult technical problems likely
- No detailed justification of estimate
- Be aware of scalability problems!

Top-Down Estimation: Delphi Method

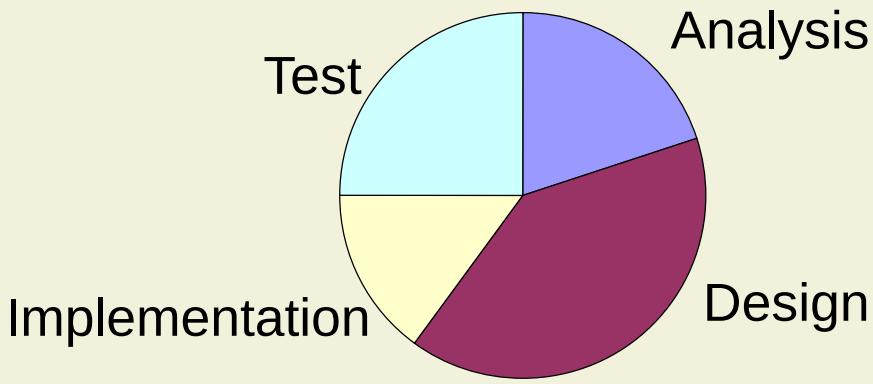


- **More accurate** than ordinary expert judgment
  - Eliminates outliers
- **More expensive** to produce

Top-Down Estimation: Typical Figures

- Typical figures for software development

- Analysis	20%
- Design	40%
- Implementation	15%
- Test	25%



- Very **helpful to validate** estimations

Bottom-Up Estimation

- Estimation by **decomposition**
  - Estimating the effort for **individual work packages**
  - Cost and accuracy depend on size of the work packages

Pros

- See “cons” of top-down estimation

Cons

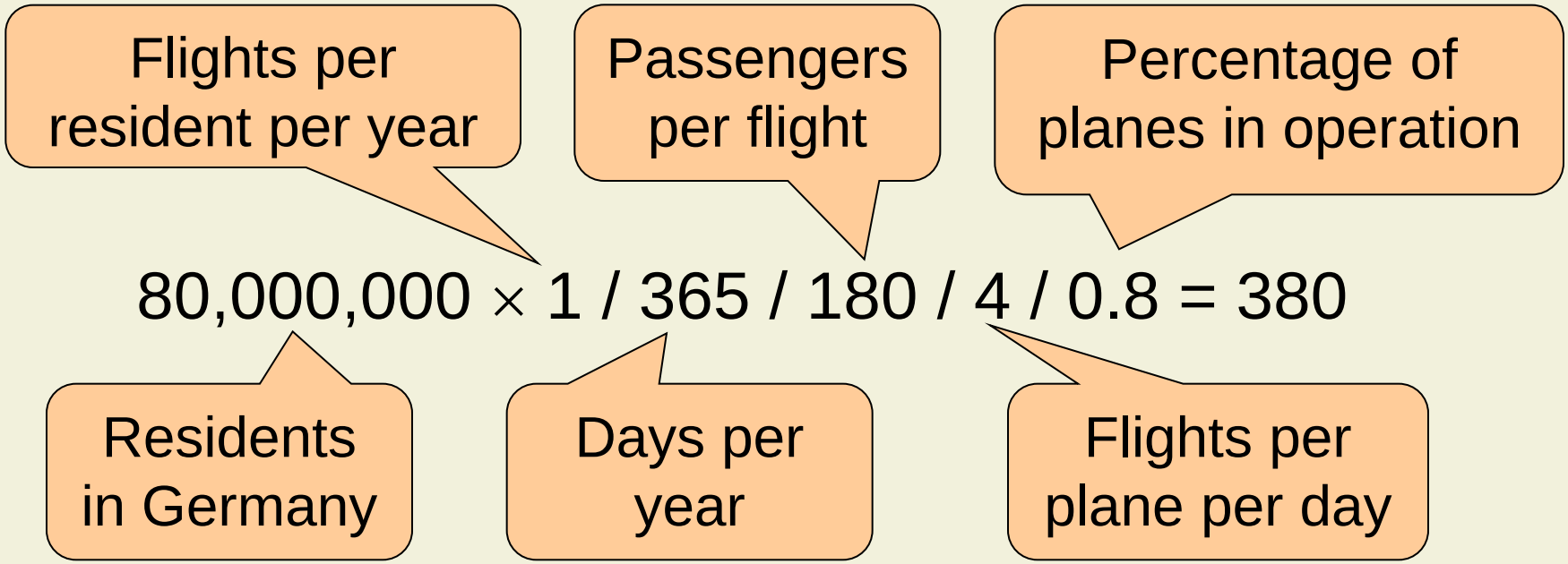
- Underestimation because effort does not grow linearly (due to complexity, etc.)
- Underestimation of integration effort
- Requires initial system design

Program Evaluation and Review Technique

- Goal: Manage probabilities with simple statistics
- Approach: Ask several experts for three estimates
  - **Optimistic, Likely (mode), and Pessimistic**
- Important formulas
  - Mean $M = (O + 4 \times L + P) / 6$
  - Deviation $V = (P - O) / 6$
- Assumptions
  - Project effort is normally distributed (more than 20 work packages)
  - Work package efforts are statistically independent (ignores single underlying cause of delay)

9. Effort Estimation

Algorithmic Estimation



- Algorithmic estimation is based on
  - **Cost model**, represented by formula
  - **Measurement of size** (passengers, destinations, etc.)
  - **Parameters** (size of planes, planes in operation, etc.)

Algorithmic Estimation of Software

- Basic cost model

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

- 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

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

- 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)

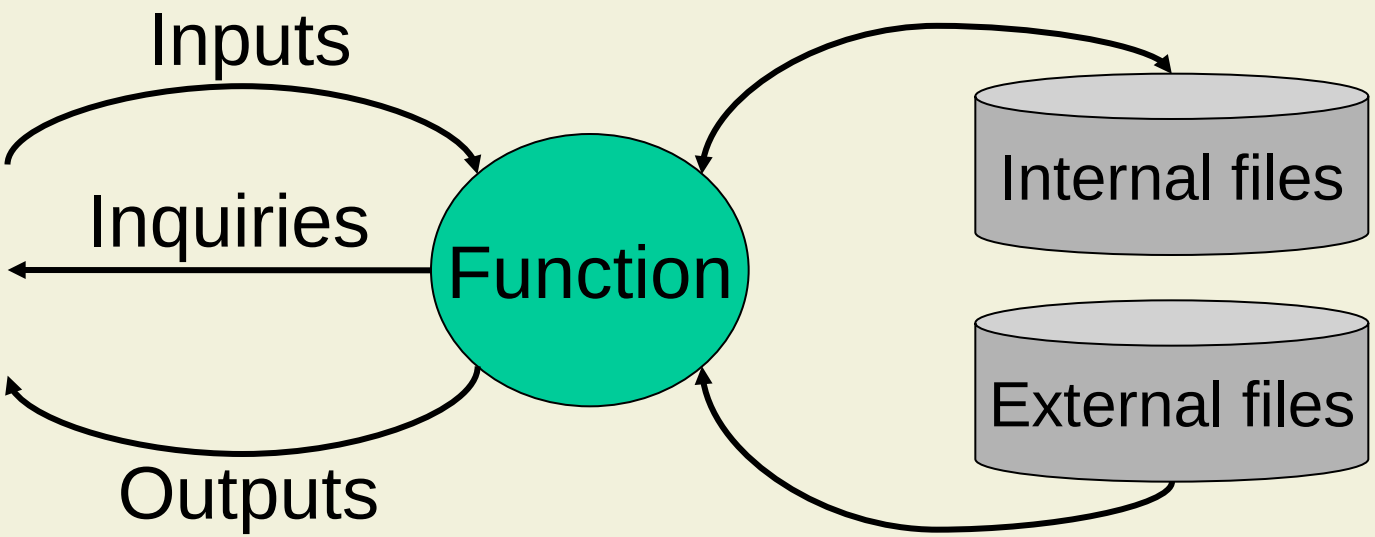
- Important examples
  - Function point analysis
  - Constructive cost model (COCOMO)

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

- Determine **complexity** of each function

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

- **Weight** each function according to complexity

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

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

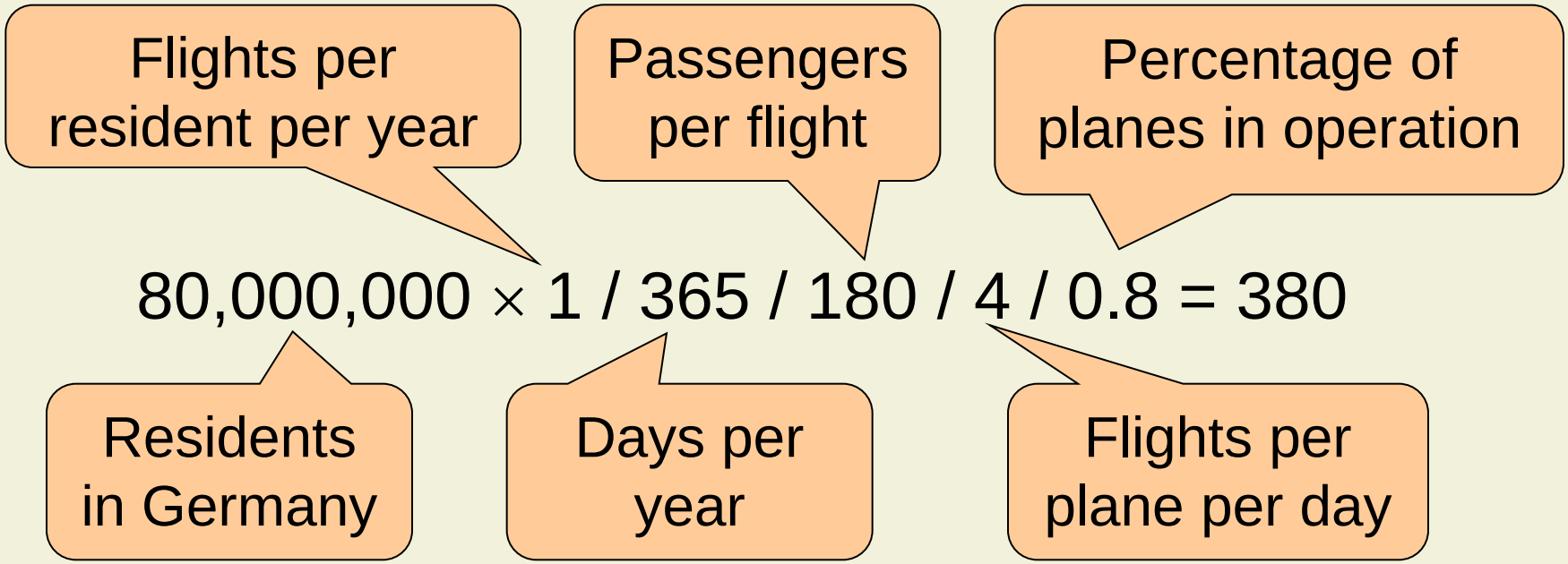
- Rate each element from 0 – 5
  - 0: no influence
  - 1: insignificant influence
  - 2: moderate influence
  - 3: average influence
  - 4: significant influence
  - 5: strong influence
- **Technical complexity factor**
  - TCF = 0.65 + 0.01 × sum
  - Varies between 0.65 and 1.35

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**

- **Adjusted function points**: FP = UFP × TCF

Calibration



- Assume that model (formula) is correct
- **Calibrate** model based on **comparable** airlines
- Estimate number of residents in the country

Determining Effort and Size

- Empirical value for effort
  - Or use a table

$$\text{Effort} = FP^{1.4} / 150$$

- Empirical value for size
- Huge differences in productivity
  - Factor 10-20 between individual programmers
  - Factor 4 between companies

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

Adaptation for OO: Use Case Points

1. Rank **actors**
   - Simple (1 point): a machine with a programmable API
   - Average (2 points): a human with a command line interface or a machine via some protocol (no API written)
   - Complex (3 points): a human with a GUI
2. Rank **use cases**
   - Simple (5 points): 3 or fewer transactions
   - Average (10 points): 4 to 7 transactions
   - Complex (15 points): 8 or more transactions
3. Calculate unadjusted use-case point count, TCF, and **adjusted use-case point** (AUCP) count
4. Compute effort: 20-30 person hours / AUCP

Constructive Cost Model (COCOMO II)

- **Three models** for three major project stages
  - Different size metrics and cost factors

Application Composition

- Used for prototypes
- Size: application points

Early Design

- Used for initial estimates based on requirements and design options
- Size: function points and language

Postarchitecture

- Used for development effort based on design specification
- Size: source LOC

Example: Early Design Model

$$\text{Effort} = 2.94 \times \text{Size}^{B} \times m(X)$$

- $B = 0.91 + 0.01 \times SF$
- SF: sum of five scale factors
  - Determined by tables
- B lies between 1.1 and 1.24

Scale factors

- Precedentedness
- Development flexibility
- Architecture / risk resolution
- Team Cohesion
- Process maturity

Example: Early Design Model (cont'd)

$$\text{Effort} = 2.94 \times \text{Size}^{B} \times m(X)$$

- m(X): product of seven cost factors
  - Determined by tables
- m(X) lies between 0.08 and 60.68

Cost factors

- Product reliability and complexity
- Development for reusability
- Platform difficulty
- Personnel capability
- Personnel experience
- Facilities
- Required development schedule

Estimation Techniques: Discussion

Empirical Estimation

- Accurate if experts are experienced
- Experts can be strongly biased (over-optimism)

Algorithmic Estimation

- Very accurate if model is calibrated
- Calibration is very difficult and expensive
- Estimation is expensive

- Empirical studies
  - Do not show that uncalibrated algorithmic estimation is, in general, more accurate
  - Show that algorithmic estimation is more accurate than experts who do not have important domain knowledge

Other Estimation Strategies

Parkinson's Law

- Work expands to fill the time available
  - Gold plating
- Effort is determined by available resources
- Important for team management

Pricing to win

- Cost is estimated to whatever the customer is willing to spend
- Common strategy to win projects
- Features are negotiated later, constrained by agreed costs
- Costs are fixed, not requirements

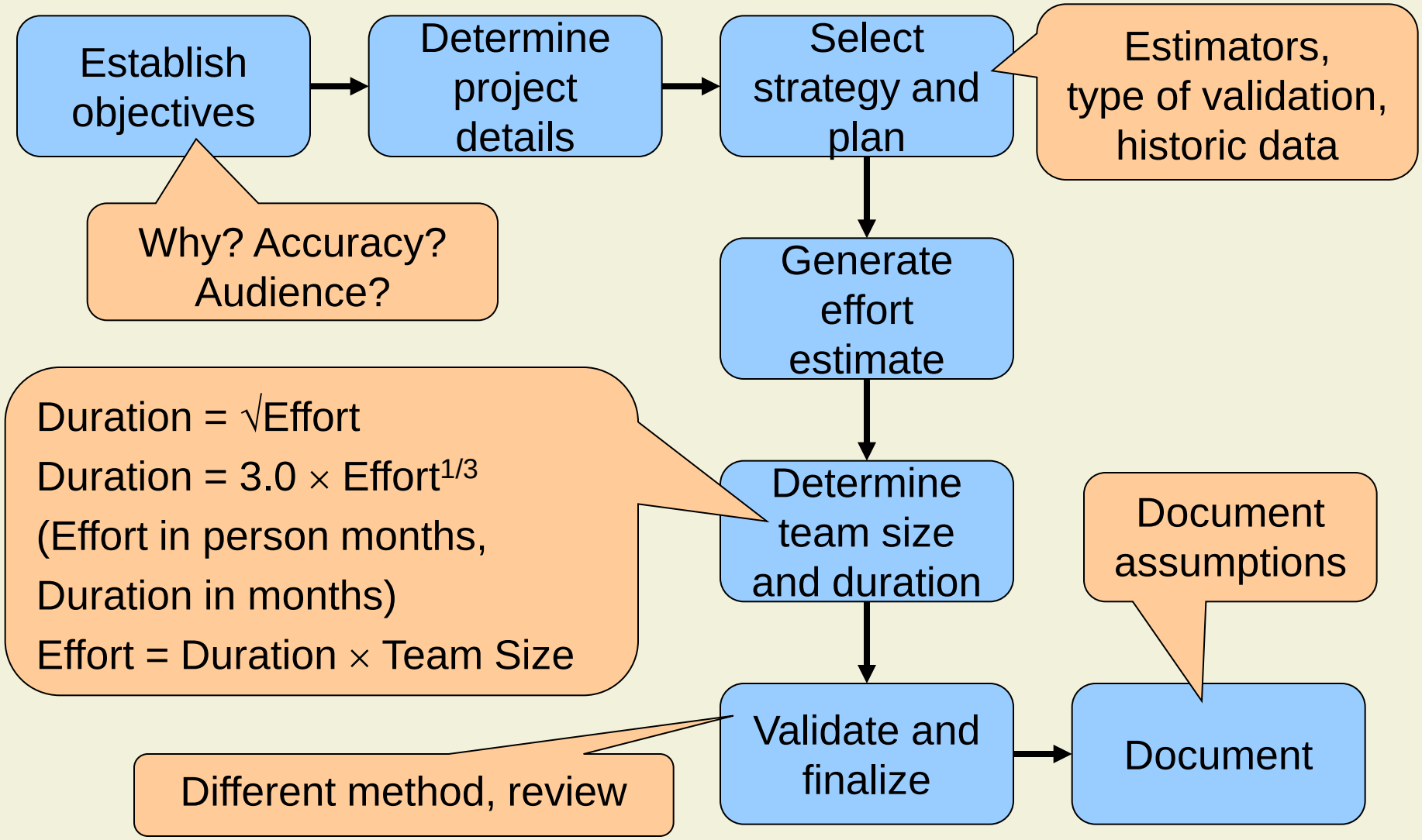
9. Effort Estimation

Estimate Types

Rough order of magnitude	Budgetary	Definitive
-25 / +75%	-10 / +25%	-5 / +10%
Initial estimates	Decision making, response to proposals	Project plan, proposals

- **Refine** your estimates at each project stage
  - Requirements document, system design, detailed design, working code

Estimating Process



Estimation Tips

- Avoid off-the-cuff estimates
- Allow time for the estimate, and plan it
- Use historic data
- Use **developer-based estimates**
- Estimate by walkthrough
- Estimate by categories
  - e.g. easy, medium, hard
- Estimate at a low level of detail
- Don’t omit common tasks (management; use **checklists**)
- Use **different techniques** and compare the results
- Change estimation practices as the project progresses

From Effort to Costs

- **Direct costs**: Costs incurred for the benefit of a specific project
  - Salaries of project staff
  - Equipment bought specifically for the project
  - Travel expenses
- **Indirect costs**: Costs incurred for the joint benefit over multiple projects (“overhead”)
  - Accounting, quality assurance department
  - Line management
  - Rooms, electricity, heating

Unit Costs

- Projects have to budget for
  - Direct costs
  - A certain share of indirect costs
- Budgets are usually determined by using unit costs
  - Unit cost: Price per unit of a resource
  - **Loaded rate**: Including indirect costs
  - **Unloaded rate**: Without indirect costs
- Examples
  - Loaded day rate for senior IT consultant: CHF 3.500
  - Loaded day rate for internal developer: CHF 1.200

From Costs to Prices

- The price is often based on the costs and a margin
- Price = Costs / (1 - Margin)
- Example
  - Costs = CHF 1.000.000
  - Margin = 5%
  - Price = CHF 1.052.632
- Price is influenced by
  - Market situation
  - Business strategy