# Measurement & Estimation

Software Engineering 2

(3103313-1)

Amirkabir University of Technology

Fall 1399-1400

RECALL

# Metrics and Measurements

Measure, Measurement, and Metrics

## Things to Measure

RECALL

**Directly Measurable (Objective)**

- Cost
- Effort
- Defect Rates
- Lines of Code (LOC)
- Pages of Doc.
- …

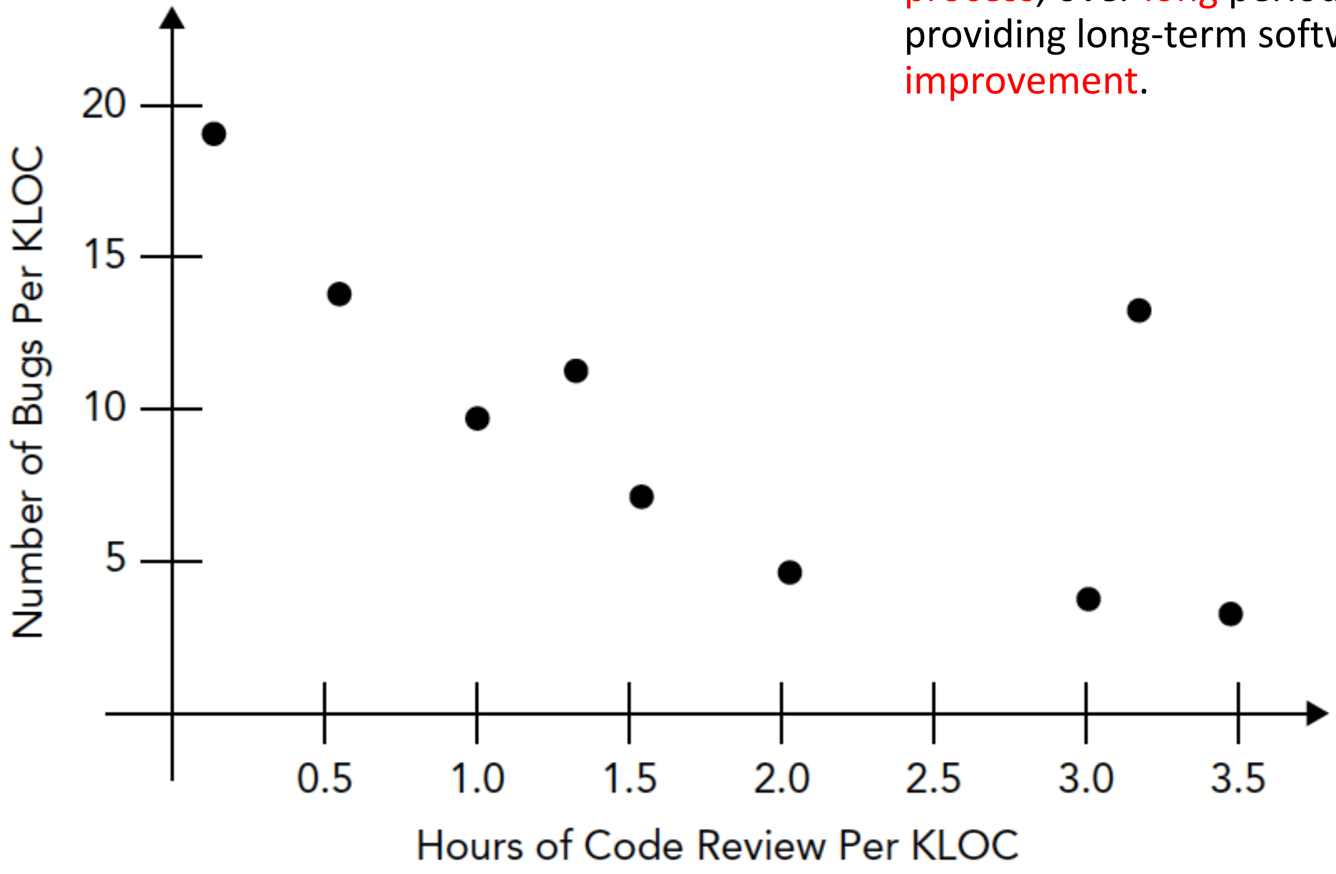
**Hard to Measure (Subjective)**

- Functionality
- Complexity
- Efficiency
- Maintainability
- Reliability
- Availability
- …

RECALL

## Process Metrics

Measure organization's development process, over long periods of time, providing long-term software process improvement.



## Product Metrics

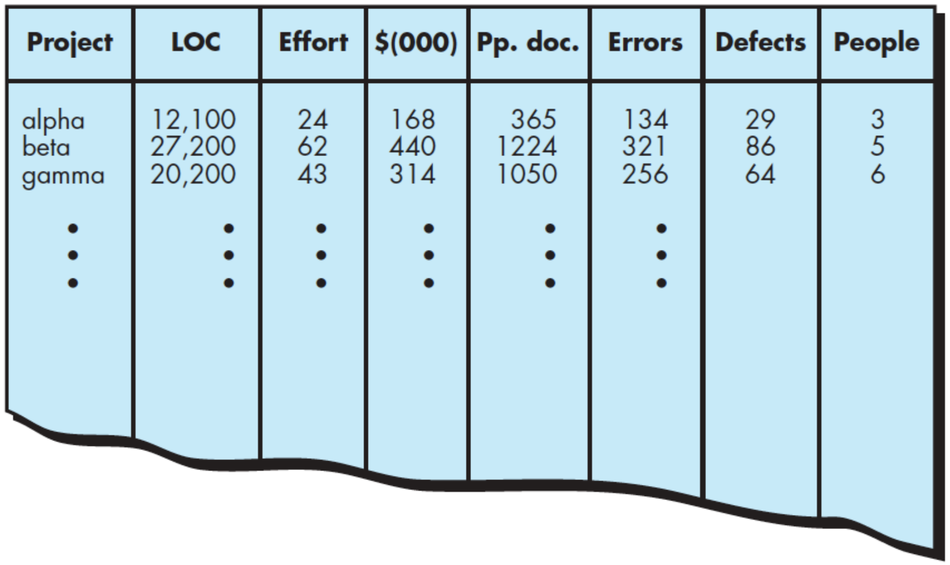
RECALL

- Predictor metrics used to quantify internal attributes of a software system.

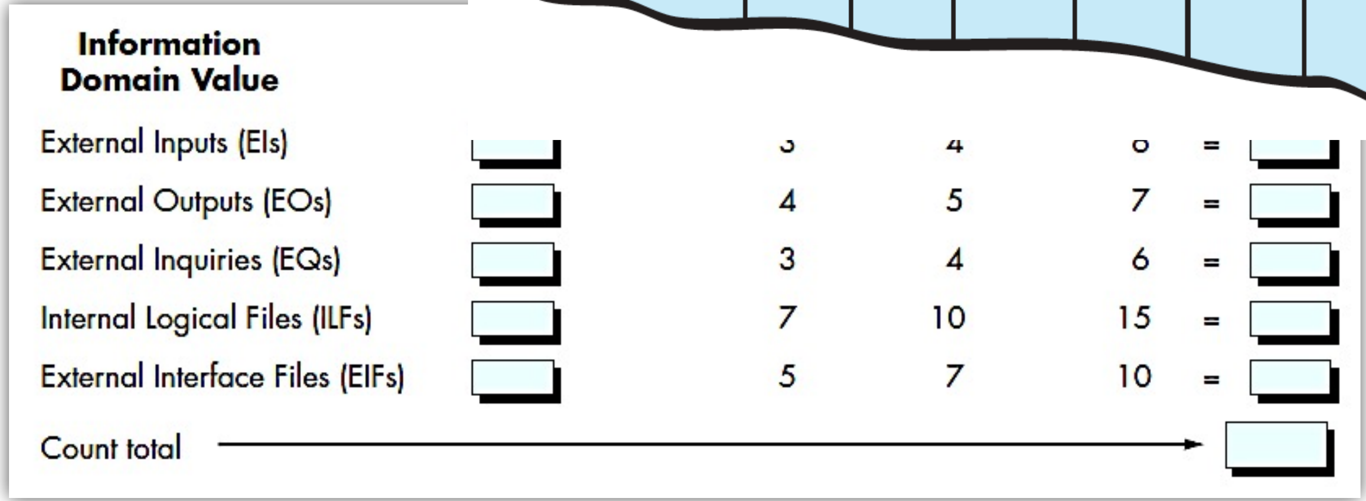
- Requirements
- Design and Architecture
- Source Code
- Tests
- …

# Product Metrics

- Size-Oriented
- Function-Oriented
- OO Metrics
- Application-Specific
    - Web, Mobile, …



| Project | LOC | Effort | $(000) | Pp. doc. | Errors | Defects | People |
|---|---|---|---|---|---|---|---|
| alpha | 12,100 | 24 | 168 | 365 | 134 | 29 | 3 |
| beta | 27,200 | 62 | 440 | 1224 | 321 | 86 | 5 |
| gamma | 20,200 | 43 | 314 | 1050 | 256 | 64 | 6 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | | |



| Information Domain Value | | | | | | |
|---|---|---|---|---|---|---|
| External Inputs (EIs) | | 3 | 4 | 6 | = | |
| External Outputs (EOs) | | 4 | 5 | 7 | = | |
| External Inquiries (EQs) | | 3 | 4 | 6 | = | |
| Internal Logical Files (ILFs) | | 7 | 10 | 15 | = | |
| External Interface Files (EIFs) | | 5 | 7 | 10 | = | |
| Count total | | | | | | |

## Product Metrics

RECALL

***Static Software Product Metrics***

- Fan-in/Fan-out
- Length of code
- Cyclomatic complexity
- Length of identifiers
- Depth of conditional nesting
- Fog index
  - average length of words and sentences in documents

***CK Object-Oriented Metrics Suite***

- Weighted methods per class (WMC)
- Depth of inheritance tree (DIT)
- Number of children (NOC)
- Coupling between object classes (CBO)
- Response for a class (RFC)
- Lack of cohesion in methods (LCOM)

## The metrics that really *do* matter

## (Agile) Process Metrics

- **Lead Time**
  - how long it takes you to go from idea to delivered software.
- **Cycle Time**
  - how long it takes you to make a change to your software system and deliver that change into production.
- **Team Velocity**
  - how many “units” of software the team typically completes in an iteration (a.k.a. “sprint”).
- **Open/Close Rates**
  - how many production issues are reported and closed within a specific time period.

## Production Metrics

- **Active days**
  - how much time a software developer contributes code to the software development project.
- **Assignment scope**
  - the amount of code that a programmer can maintain and support in a year.
- **Efficiency**
  - the amount of productive code contributed by a software developer.
- **Code churn**
  - the number of lines of code that were modified, added or deleted in a specified period of time.
- **Impact**
  - the effect of any code change on the software development project.

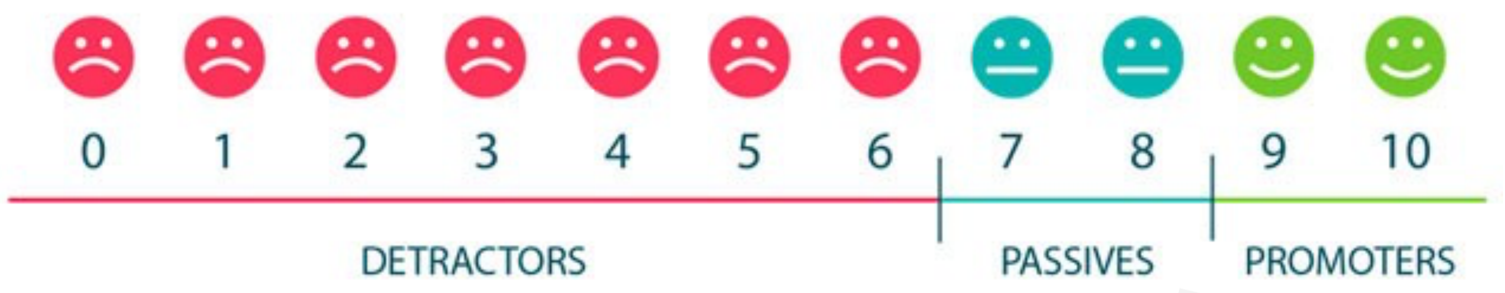
## Operational Metrics

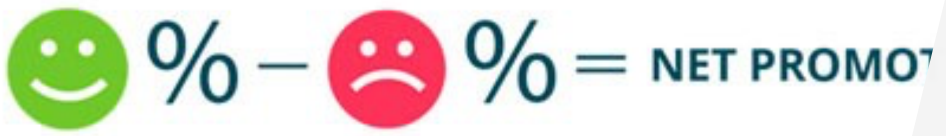
- How software is running in production and how effective operations staff are at maintaining it.

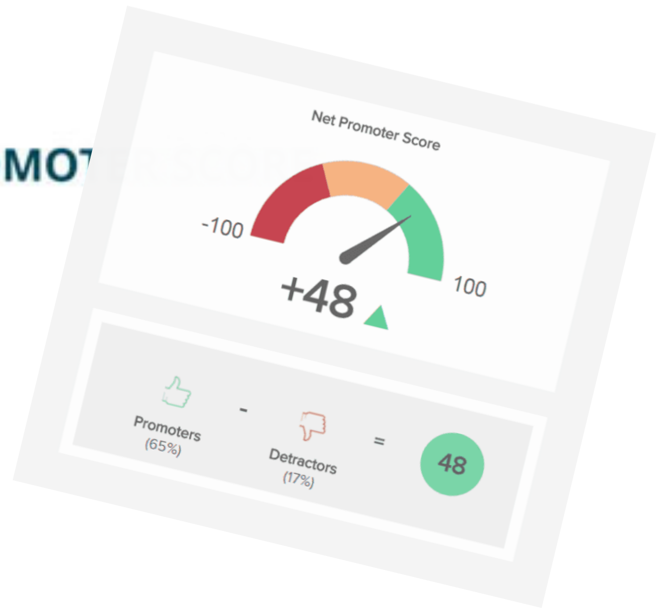
- **Mean time between failures (MTBF)**
- **Mean time to recover/repair (MTTR)**

- **Application crash rate**
  - how many times an application fails divided by how many times it was used.

## Customer Satisfaction







- **Net Promoter Score (NPS)**
- **Customer Satisfaction Score (CSAT)**
- **Customer Effort Score (CES)**

## Other Metrics

- Size-oriented metrics
  - Errors per KLOC
  - Defects per KLOC
  - Cost per KLOC
- Function-oriented metrics
  - Errors per FP or Defects per FP
- Test Metrics
- Security Metrics
- Defect Removal Efficiency (DRE)

- **Product KPIs**
  - Customer Lifetime Value (CLTV or LTV)
  - Customer Acquisition Cost (CAC)
  - Daily/Monthly Active User ratio
  - Session duration
  - Traffic (paid/organic)
  - Bounce rate
  - Retention rate
  - Churn rate
  - Number of sessions per user
  - Number of user actions per session

## Appropriate use of software metrics

four guidelines for an appropriate use of software metrics, by Patrick Kua.

- Link software metrics to goals
- Track trends, not numbers
- Set shorter measurement periods
- Stop using software metrics that do not lead to change

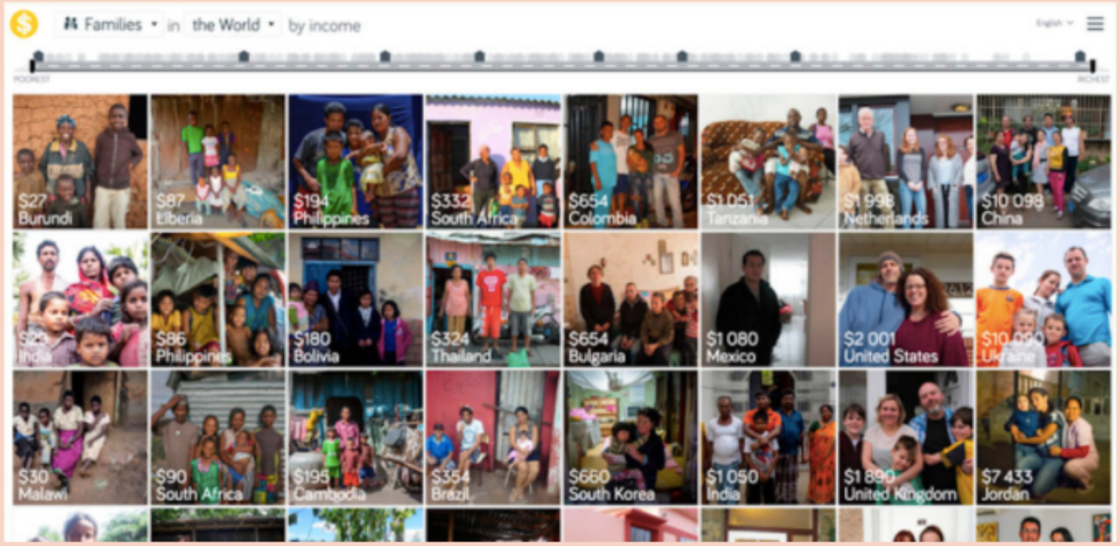
- How to measure productivity?
- How to measure effectiveness?
- Outcome vs. Output?

## break

## Dollar Street

Visit hundreds of homes on all income levels across the world, to complement the media's skewed selection of dramatic images of other places.

See the reality behind the data



**Gapminder** is an independent educational non-profit fighting global misconceptions.

https://www.gapminder.org/

## Estimation

**"… estimation is valuable when it helps you make a significant decision**."

Martin Fowler

## Estimation
## Risk and Uncertainty

Estimation requires

- Experience
- Access to historical information (metrics)
- Quantitative predictions over qualitative information

- Project Complexity
- Project Size
- Degree of Structural Uncertainty
- Availability of Historical Info.
- …

Estimation and modern software development approaches?

- Not become obsessive about estimation; it most probably goes wrong!
- Revisit the estimate, as more information is known

## Software Project Estimation

*… from a black art to a series of systematic steps …*

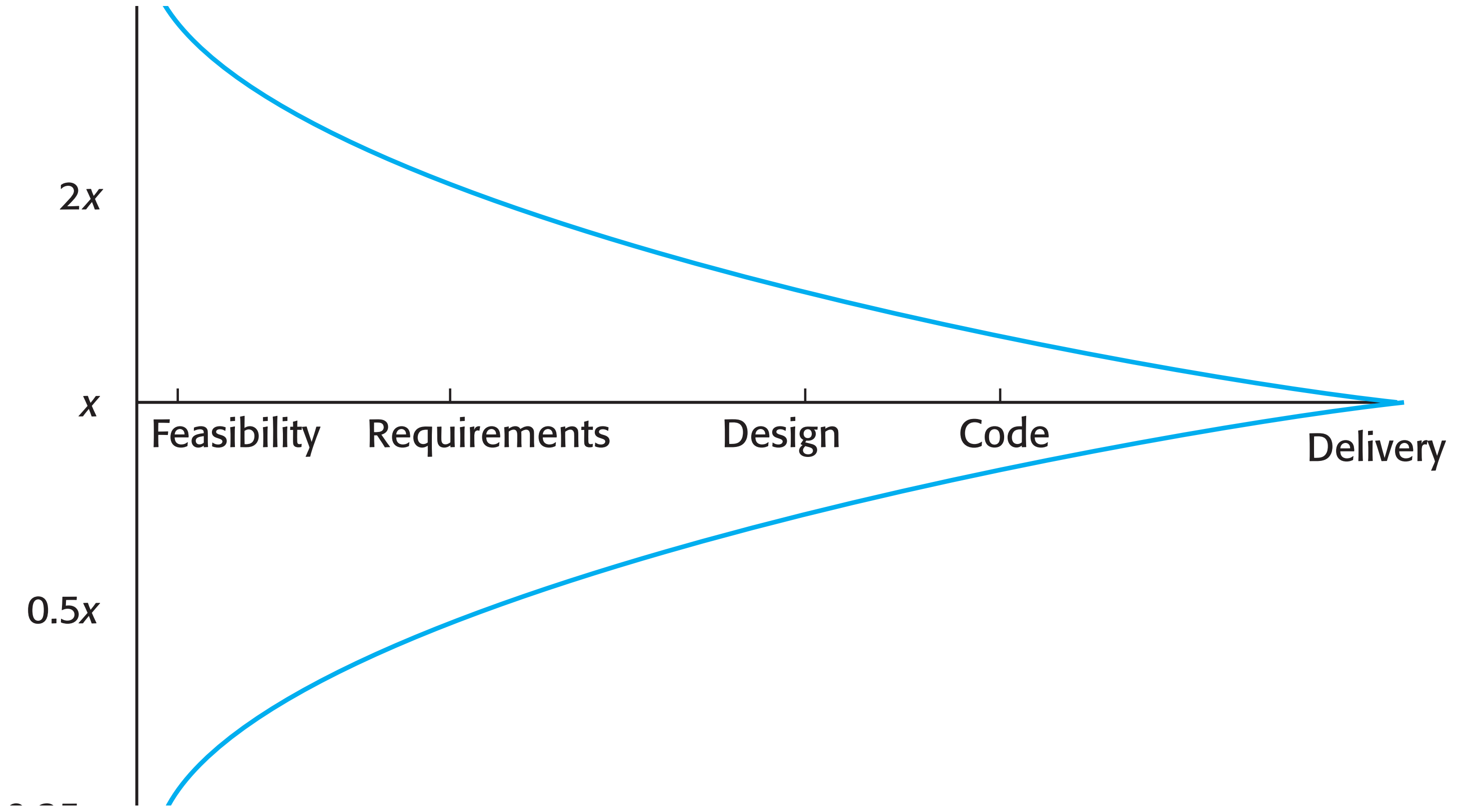
- Delay estimation(???!!!)
- Base estimates on similar projects
- Use decomposition techniques
    - e.g., software sizing, problem-based, process-based, use case-based …
    - Productivity Metrics (?)

$$S = \frac{s_{\text{opt}} + 4s_m + s_{\text{pess}}}{6}$$

Three-point or Expected-value Estimate

- Use empirical models

## Estimate Uncertainty



# Empirical Estimation Models

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

- Matson, J., et al (1994) “Software Cost Estimation Using Function Points,” IEEE Trans. Software Engineering

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

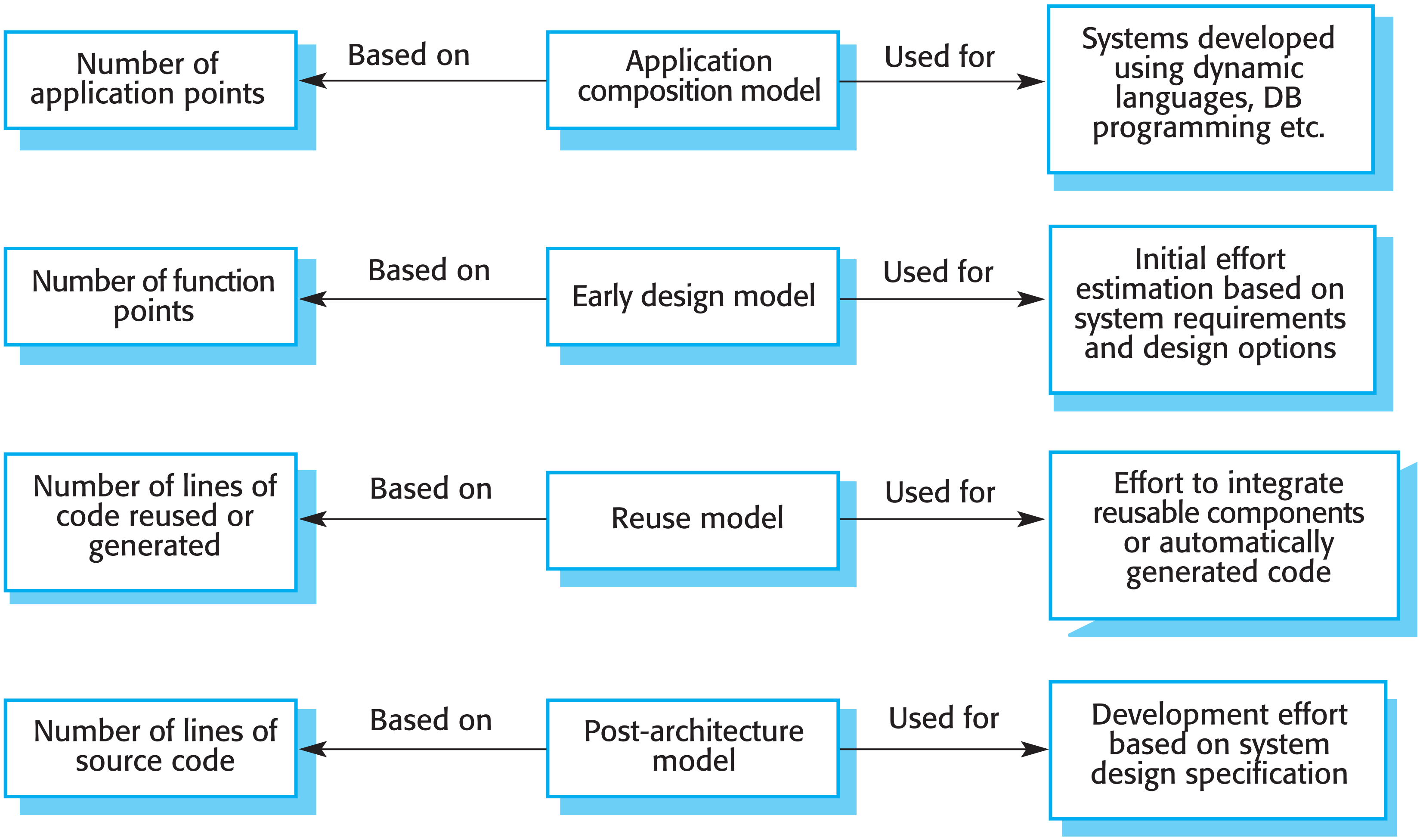
- COCOMO II Model

  (Constructive Cost Model)
  - Multi-level models: composition model, early design, reuse model, post-architecture
  - Considers different software dev. approaches, e.g., reuse

- Software Equation

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

# COCOMO Estimation Model


Number of application points
Based on
Application composition model
Used for
Systems developed using dynamic languages, DB programming etc.
Number of function points
Based on
Early design model
Used for
Initial effort estimation based on system requirements and design options
Number of lines of code reused or generated
Based on
Reuse model
Used for
Effort to integrate reusable components or automatically generated code
Number of lines of source code
Based on
Post-architecture model
Used for
Development effort based on system design specification

## More?