CTIS359

Principles of Software Engineering

Software Cost Estimation: Algorithmic and Non-Algorithmic Est. Techniques COCOMO (Constructive Cost Model(s))

"Every line of code costs money to write and more money to support. It is better for the developers to be surfing than writing code that won't be needed."

Mary Poppendieck

Today

- What is Software Cost Estimation?
- Categorization of Software Cost Estimation Techniques
  - Expert Judgment
  - Parkinson's Law
  - Pricing-to-Win
  - Top-Down-Estimation & Bottom-Up Estimation
  - Analogy based
  - Algorithmic Models
    - Cost Models (Empirical Factor Models)
      - Ex: COCOMO I & II
    - Constraint Models

What is Software Cost Estimation?

- Software cost estimation is the *process of predicting the effort required to develop a software system and time to develop it*.
- Models and other approaches have been used.
- Models provide mathematical algorithms to compute cost as a function of **variables** such as size (LOC, FP) and/or cyclomatic complexity, etc.

Source: https://people.ucalgary.ca/~far/Lectures/SENG421/PDF/SENG421-07.pdf

Software Cost/Schedule Estimation

- Models may be classified into 2 major categories:
  - Algorithmic and Non-algorithmic
  - Each has its own strengths and weaknesses.
- A key factor in selecting a cost estimation model is the **accuracy** of its estimates.
- Unfortunately, despite the large body of experience with estimation models, the accuracy of these models is not satisfactory.

Source: Handbook of Software Engineering and Knowledge Engineering, pp. 307-324 (2002)

Accuracy vs. Precision

Accuracy vs. Precision

doğruluk, kesinlik

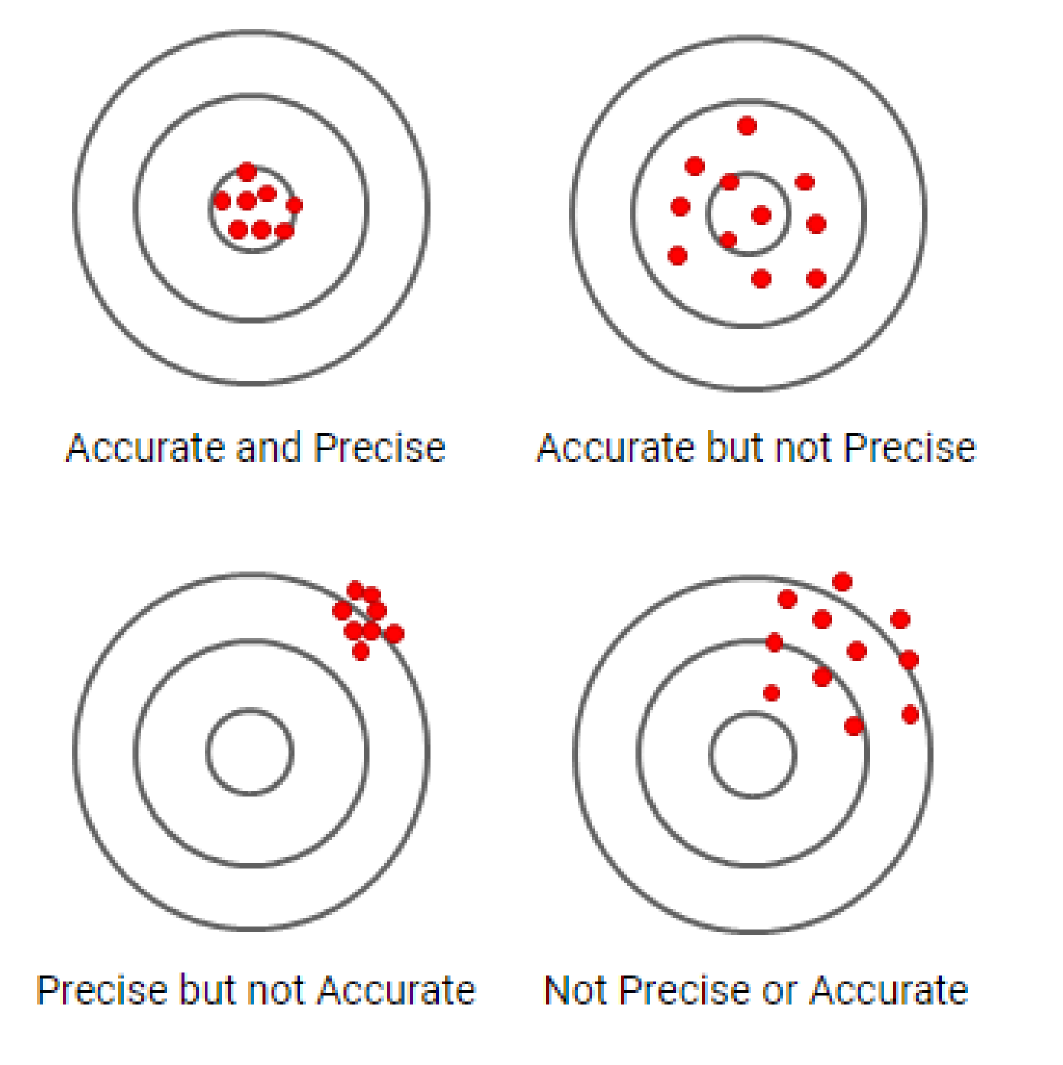
hassasiyet

accurate: conforming exactly to truth or to a standard : EXACT
accurate: going to, reaching, or hitting the intended target : not missing the target

precision: the degree of refinement with which an operation is performed or a measurement stated

Source: https://www.merriam-webster.com/

Accuracy vs. Precision



Source: https://www.medcalc.o

Good Estimation ☺

- Good estimates allow the following:
  - Accurate calculation of the project cost and its feasibility
  - Accurate scheduling of the project
  - Measurement of progress and costs against the estimates
  - Determining the resources required for the project

Source: Concise Guide to Software Engineering From Fundamentals to Application Methods, O'Regan Gerard, 2017

Poor Estimation ☹

- Poor estimation leads to:
  - Projects being over- or underestimated
  - Projects being over or under-resourced (impacting staff morale)
  - Negative impression of the PM

Software Cost Estimation

- Algorithmic Models cost can be sub-categorized as:
  - Cost Model:
    - Ex: COCOMO and its variants
  - Constraint Model:
    - SLIM
      - SLIM (**S**oftware **LI**fecycle **M**anagement) is the name given by Putnam to the proprietary suite of tools his company QSM, Inc.
  - Most of these models are available as automated tools.
  - Many estimation models have been around +40 years.

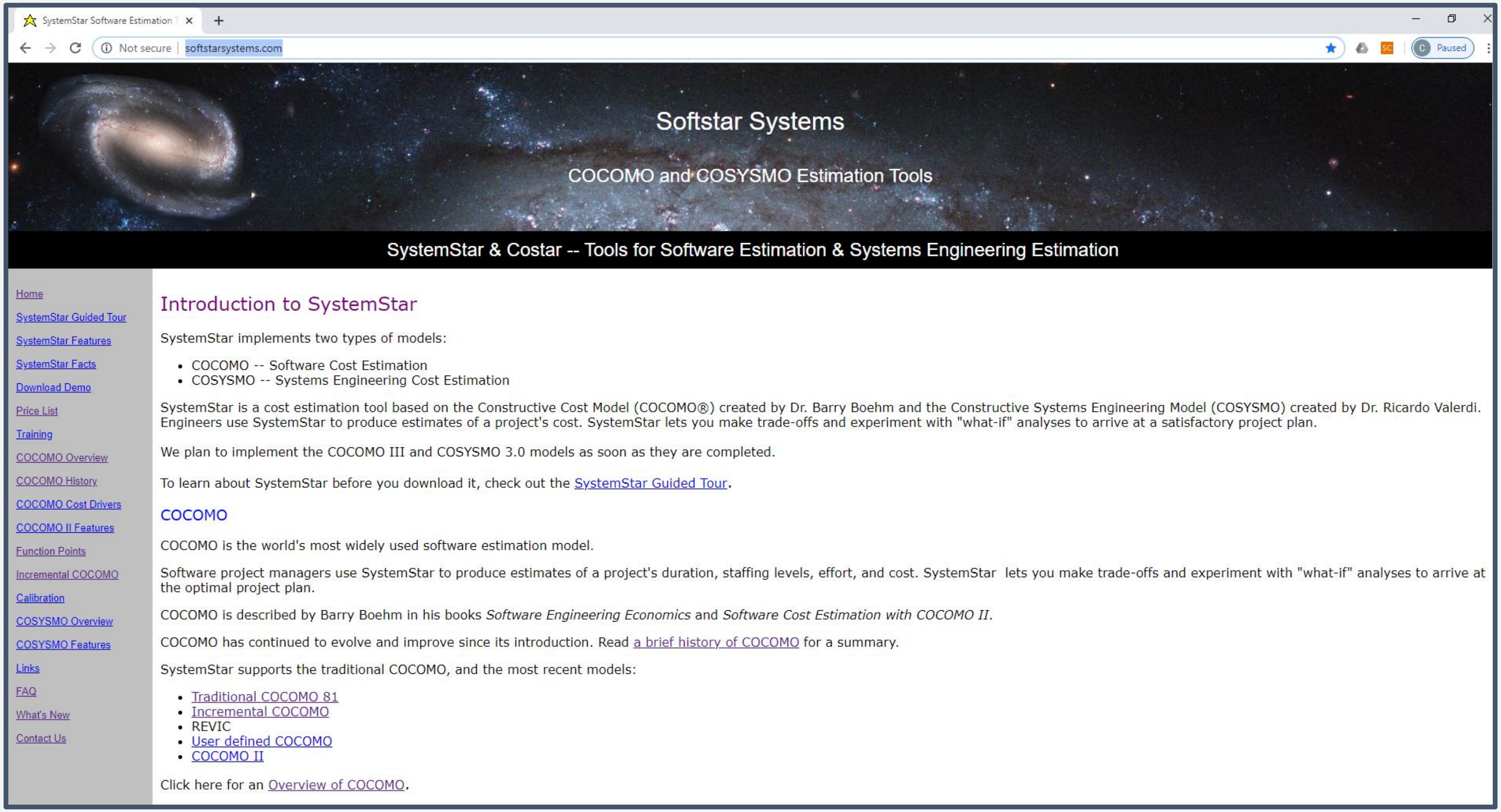
Source: https://people.ucalgary.ca/~far/Lectures/SENG421/PDF/SENG421-07.pdf

Software Cost Estimation

- Algorithmic Models cost can be sub-categorized as:
  - Cost Model:
    - Ex: COCOMO and its variants
  - Constraint Model:
    - SLIM
      - SLIM (**S**oftware **LI**fecycle **M**anagement) is the name given by Putnam to the proprietary suite of tools his company QSM, Inc.
  - Most of these models are available as automated tools.
  - Many estimation models have been around +40 years.

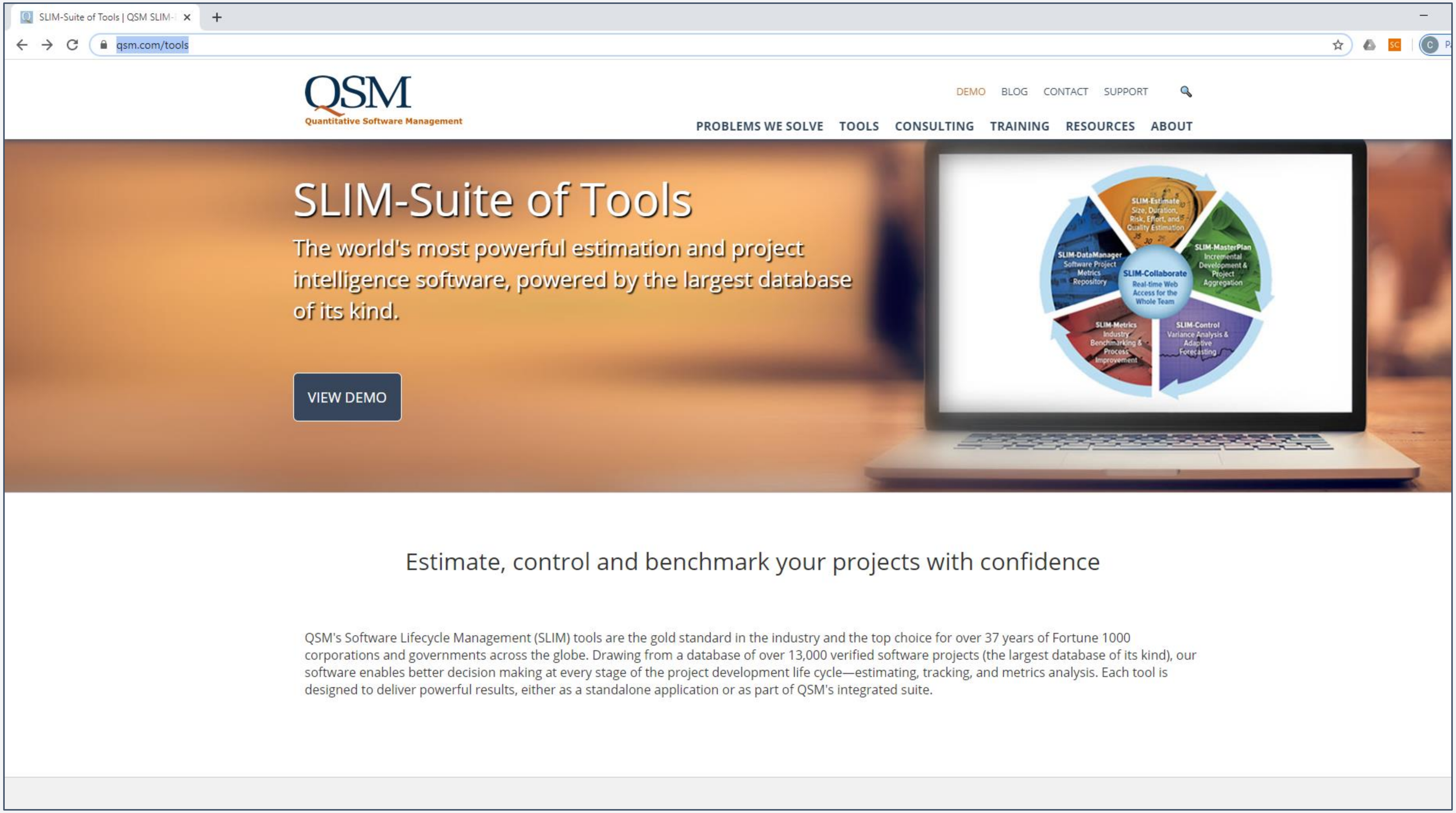
Source: https://people.ucalgary.ca/~far/Lectures/SENG421/PDF/SENG421-07.pdf

Automated Tools



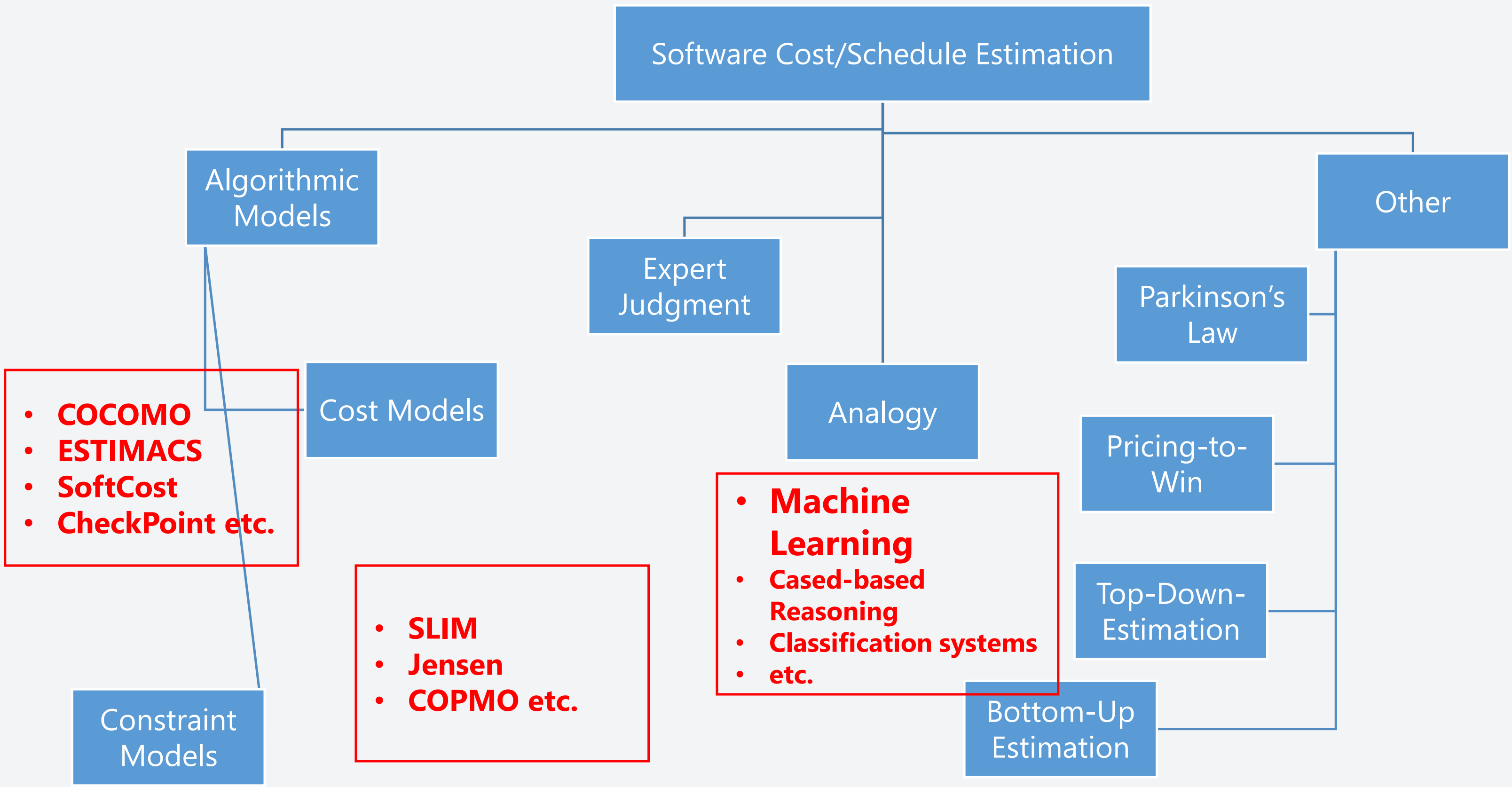
Source: http://www.softstarsystems.com/

Automated Tools



Source: https://www.qsm.com/tools

Software Cost Estimation - Techniques



Source: https://people.ucalgary.ca/~far/Lectures/SENG421/PDF/SENG421-07.pdf

Parkinson's Law

- Parkinson's Law states that *work expands to fill the time available*.
  - This means that cost is determined by **available resources** rather than objective assessment.
  - **Ex:** If the software to be delivered in 12 months and 5 people are available, the effort required is estimated to be 60 person-month.
- Advantage:
  - No overspending ☺
- Disadvantage:
  - System is usually unfinished or effort is wasted ☹

Source: https://people.ucalgary.ca/~far/Lectures/SENG421/PDF/SENG421-07.pdf

Pricing-to-Win

- Software cost/schedule is estimated to be whatever *the customer has available to spend on the project*.
- The estimated effort depends on the customer's budget and NOT on the software functionality.
- Advantage:
  - Good chances to win the contract ☺
- Disadvantage:
  - The probability that cost accuracy reflect the work required and the probability that the customer gets the system s/he wants are low ☹

Source: https://people.ucalgary.ca/~far/Lectures/SENG421/PDF/SENG421-07.pdf

Top-Down-Estimation

- A cost/schedule estimate is established by considering the overall functionality of the product and how that functionality is provided by interacting sub-functions.
- Cost estimates are made on the basis of the logical function rather that the components implementing that function.
- Advantage:
  - Takes into accounts costs such as integration and CM, documentation. ☺
- Disadvantage:
  - Can underestimate the cost of solving difficult low-level technical problems. ☹

Source: https://people.ucalgary.ca/~far/Lectures/SENG421/PDF/SENG421-07.pdf

Bottom-Up Estimation

- Starts at the lowest system level.
- The cost/schedule of **each component** is estimated. All these costs are added to produce a final cost estimate.
- Advantage:
  - Can be accurate, if the system has been designed in detail ☺
- Disadvantage:
  - May underestimate costs of system level activities☹
    - such as integration and documentation

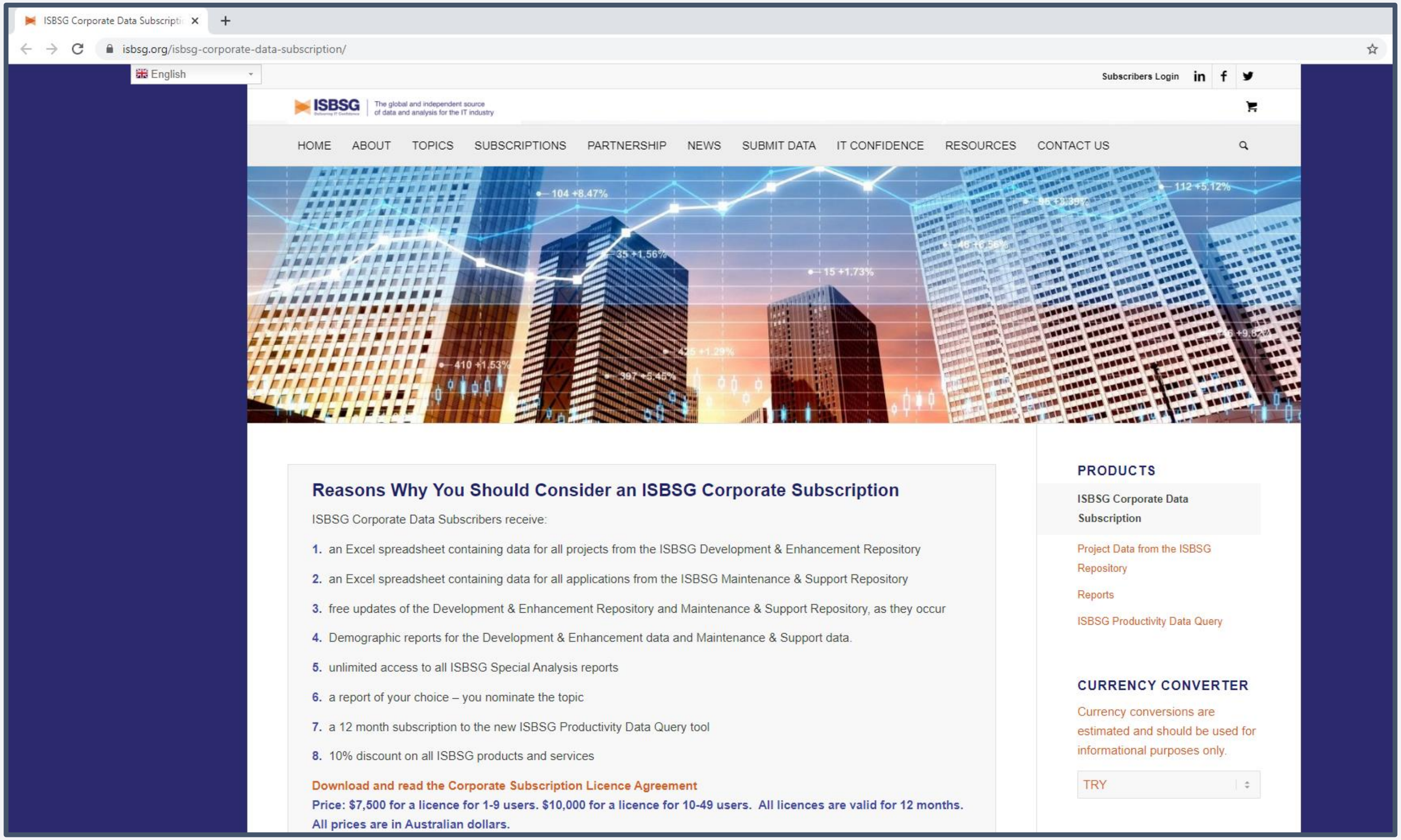
Source: https://people.ucalgary.ca/~far/Lectures/SENG421/PDF/SENG421-07.pdf

Analogy

- This technique is applicable when other projects is the same domain have been done in the past.
- The cost of the new project is computed by comparing the project to a similar project in the same domain.
- Advantage:
  - Accurate if **similar** projects are available
- Disadvantages:
  - Impossible if no comparable project available. ☹
  - Needs systematically maintained project database (Expensive) ☹
  - It is not always clear how to define a good similarity function. ☹

Source: https://people.ucalgary.ca/~far/Lectures/SENG421/PDF/SENG421-07.pdf

ISBSG (International Software Benchmarking Standards Group) is a not-for-profit organization



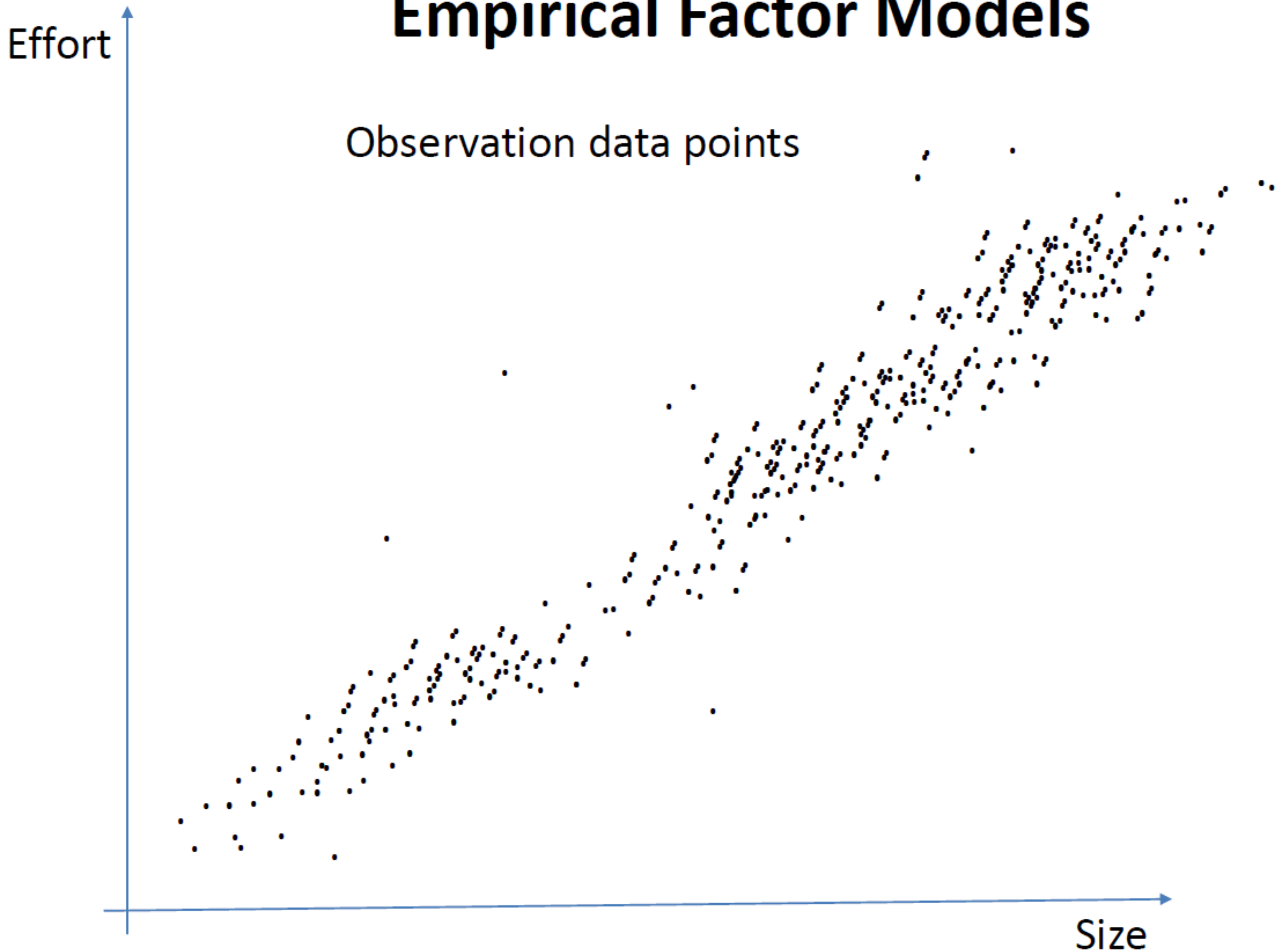
Source: https://www.isbsg.org/

Algorithmic Models

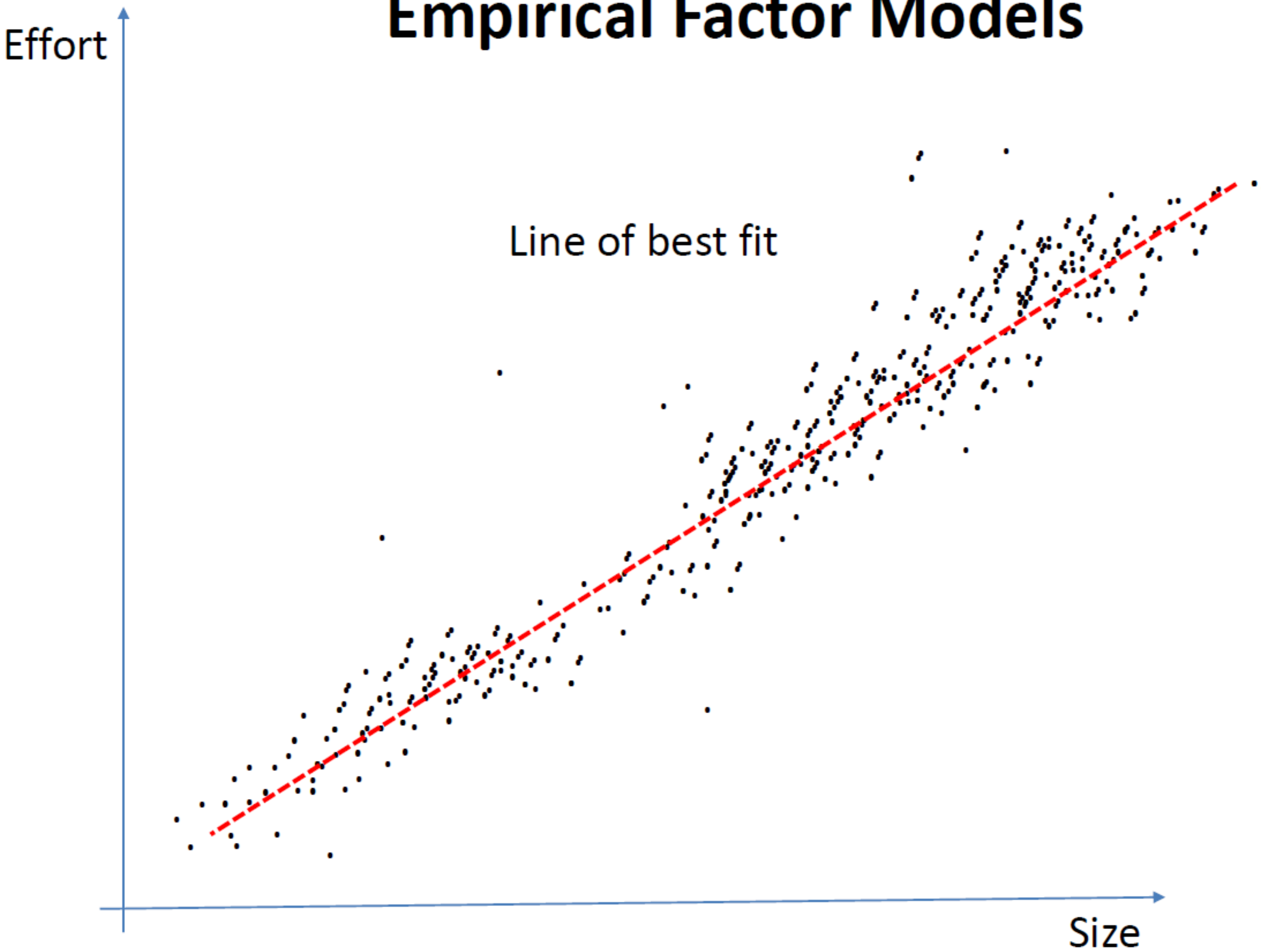
- Uses historical cost information and relates it some software project measure(s)
- Two main categories (Kitchenham, 1991)
  - **Empirical Factor Models (Cost Models)**
    - Provides estimates of project cost (effort) and schedule based on the size; **derived from the empirical data**.
  - **Constraint Models**
    - Assume a model of complex relationship between cost (effort), schedule, size, i.e., assume that productivity is not only a function of effort (input) AND size (output), but that it is "constrained" by the (anticipated) schedule and/or complexity.

Source: https://people.ucalgary.ca/~far/Lectures/SENG421/PDF/SENG421-07.pdf

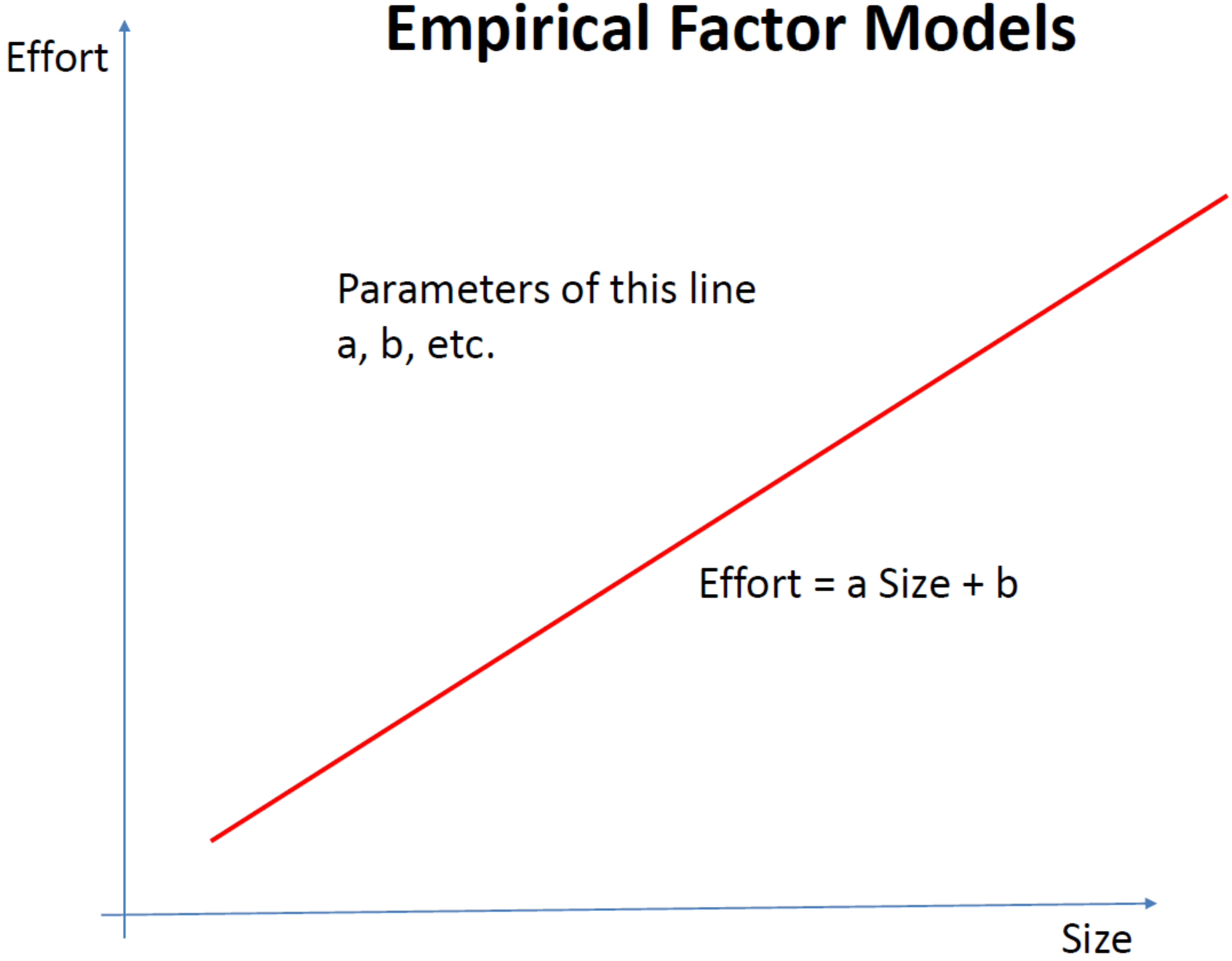
Empirical Factor Models



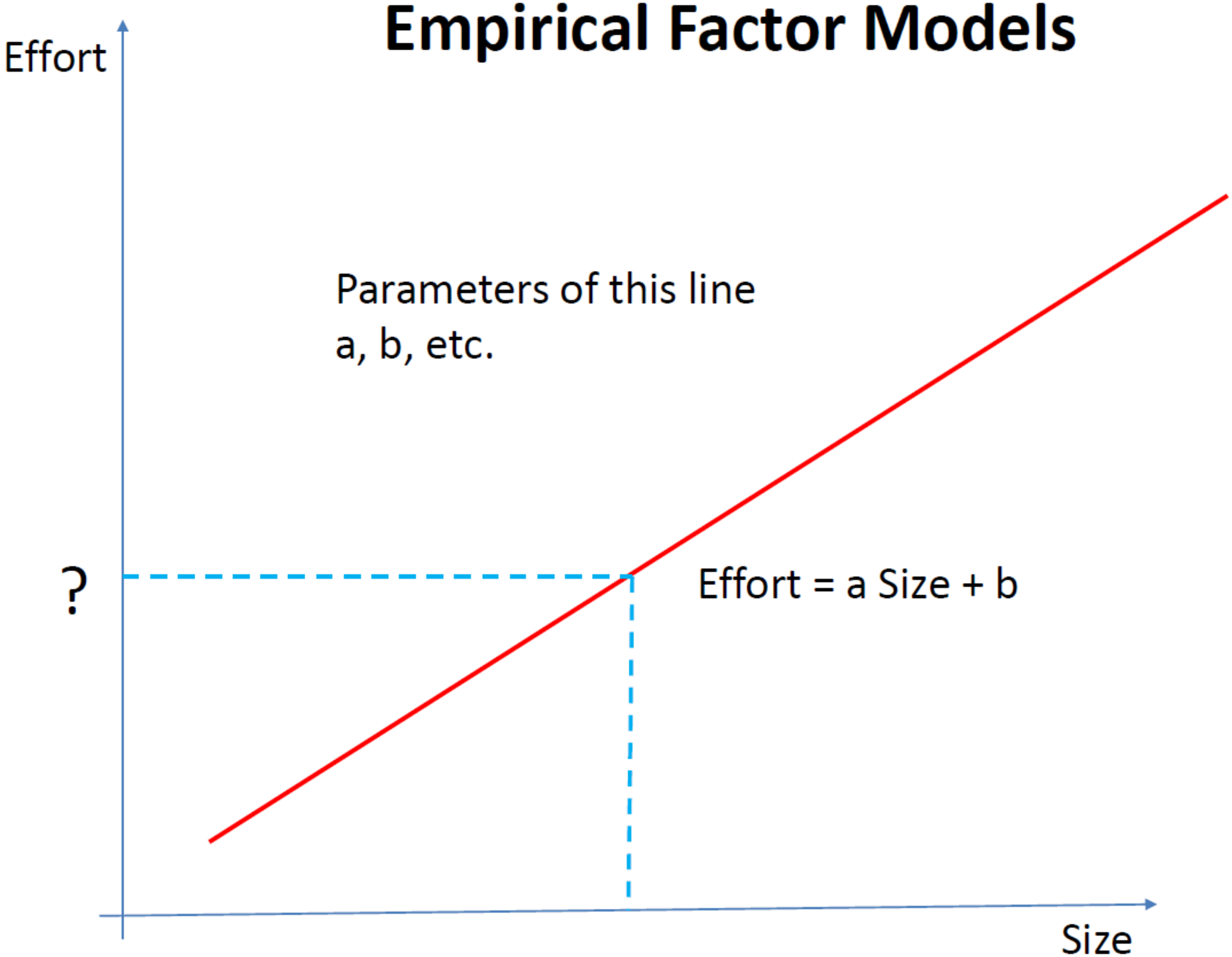
Empirical Factor Models



Empirical Factor Models



Empirical Factor Models



Algorithmic Models

- Uses historical cost informa... e software project measure(s)
- Two main categories (Kitche...
  - **Empirical Factor Models (C...**
    - Provides estimates of project c... on the size; derived from the emp...
  - **Constraint Models**
    - Assume a model complex relat... schedule, size, i.e., assume tha... ion of effort (input) and size (output), but that it is "constrained" by the (anticipated) schedule and/or complexity.

Source: https://people.ucalgary.ca/~far/Lectures/SENG421/PDF/SENG421-07.pdf

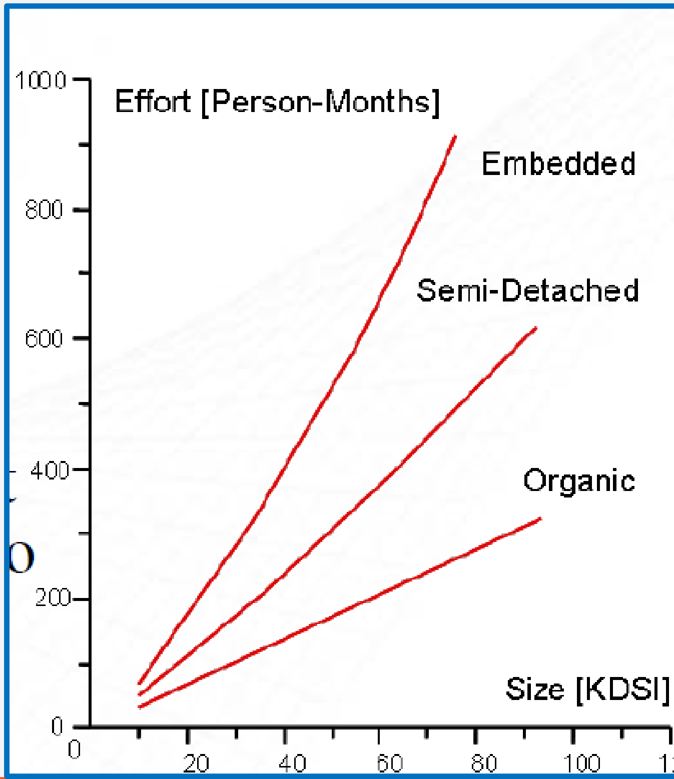
Empirical Factor Models (a.k.a. Cost Models)

- Cost models provide direct estimates of effort.
- Cost models typically are based on:
  - a primary cost factor such as size
  - a # of secondary adjustment factors or cost drivers
- Cost drivers are characteristics of the project, process, product, and resources that influence effort.
- Cost drivers are used to adjust the preliminary estimate provided by the primary cost factor.

Source: https://people.ucalgary.ca/~far/Lectures/SENG421/PDF/SENG421-07.pdf

Empirical Factor Models (a.k.a. Cost Models)

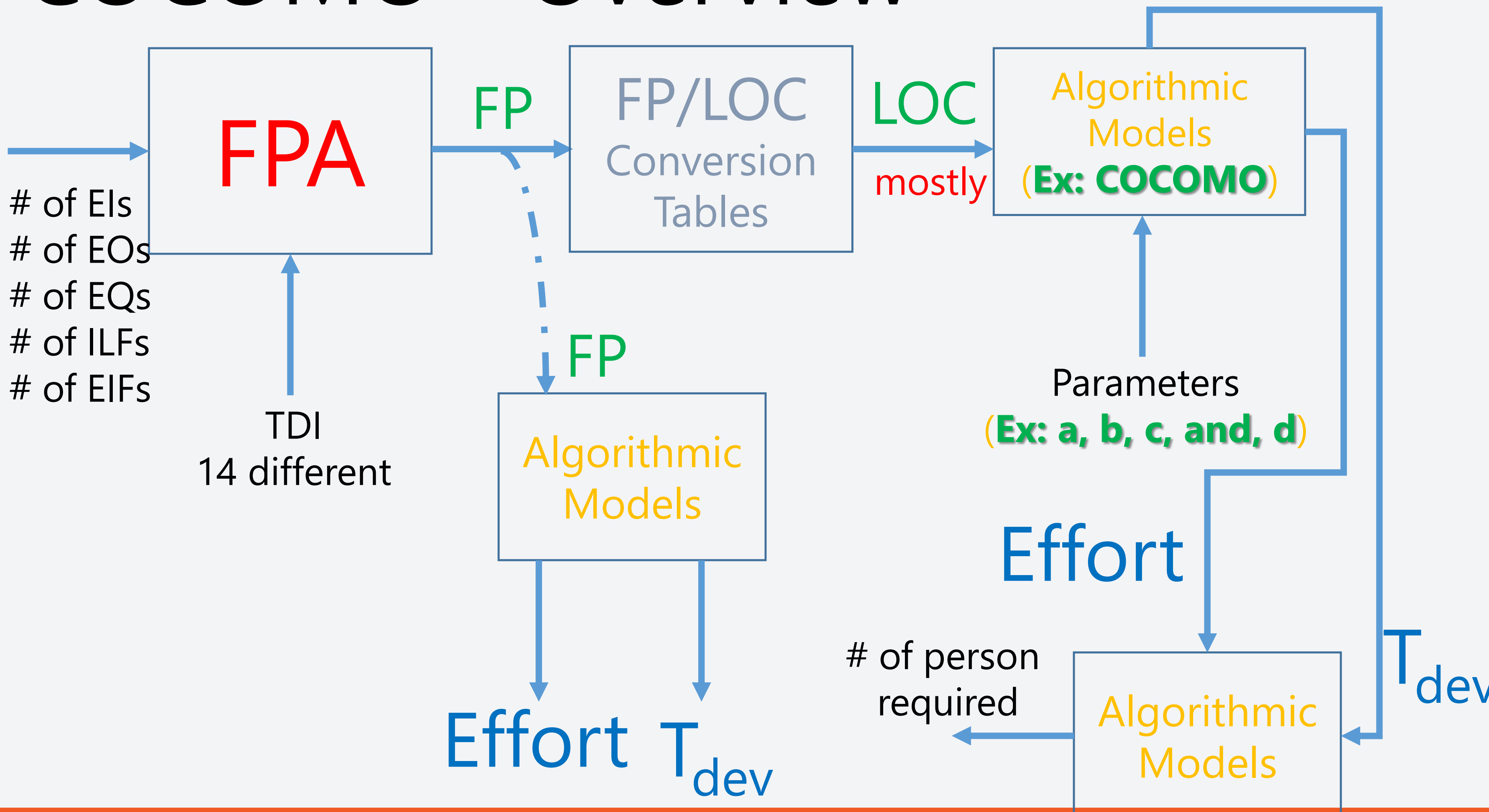
- Effort is plotted against the primary cost factor (usually LOC or FP) for a series of projects.
- The line of best fit is calculated for data points.
- If the primary cost factor were a perfect predictor of effort, then every point on the graph would lie on the line of best fit.
- In reality, there is usually significant residual error. Therefore necessary to identify the factors that cause variation between predicted and actual effort. These parameters are added to the model as *cost drivers*.



Source: https://people.ucalgary.ca/~far/Lectures/SENG421/PDF/SENG421-07.pdf

- Boehm's [1981] definition of organic, semidetached, and embedded systems:
- **Organic:** A development project can be considered of organic type, if the project deals with developing a well understood application program, the size of the development team is reasonably small, and the team members are experienced in developing similar types of projects.
- **Semidetached:** A development project can be considered of semidetached type, if the development consists of a mixture of experienced and inexperienced staff. Team members may have limited experience on related systems but may be unfamiliar with some aspects of the system being developed.
- **Embedded:** A development project is considered to be of embedded type, if the software being developed is strongly coupled to complex hardware, or if the stringent regulations on the operational procedures exist.

COCOMO - Overview



COCOMO

- The word "constructive" implies that the complexity of the model can be understood because of the openness of the model, which permits exactly to know WHY the model gives the estimates it does.
- First published by Dr. Barry Boehm in 1981, and reflected the software development practices of these days.
  - Since this time many efforts were done in the improvement of the software development techniques.
  - Some of the changes were
    - moving away from mainframe overnight batch processing to real time applications
    - difficulty in effort in building software for reusing
    - new kind of system development in including COTS components and
    - spending as much effort on designing and managing the software development process as was once spent creating the software product

Source: COCOMO (Constructive Cost Model) Seminar on Software Cost Estimation, presented by Nancy Merlo – Schett

Background of COCOMO

- 63 projects size range from 2,000 SLOC to 966,000 SLOC
- It drew on a study of 63 projects at TRW Aerospace where Boehm was Director of Software Research and Technology.
- Effort is measured in staff months (19 days per month or 152 working hours per month)

COCOMO II

- These changes urged to revise the existing model.
- By the joint efforts of USC-CSE (University of California, Center for Software Engineering) and the COCOMO II Project Affiliate Organizations, the COCOMO II model was presented.
- COCOMO II Program Affiliates:
  - Aerospace, Air Force Cost Analysis Agency, Allied Signal, DARPA, DISA, Draper Lab, EDS, E-Systems, FAA, Fidelity, GDE Systems, Hughes, IDA, IBM, JPL, Litton, Lockheed Martin, Loral, Lucent, MCC, MDAC, Microsoft, Motorola, Northrop Grumman, ONR, Rational, Raytheon, Rockwell, SAIC, SEI, SPC, Sun, TASC, Teledyne, TI, TRW, USAF Rome Lab, US Army Research Labs, US Army TACOM, Telcordia, and Xerox.

Source: COCOMO (Constructive Cost Model) Seminar on Software Cost Estimation, presented by Nancy Merlo – Schett

Estimating Effort and Duration from LOC

- Once LOC have been estimated, either
  - through use of historical data OR
  - by comparison to related projects OR
  - via FPs

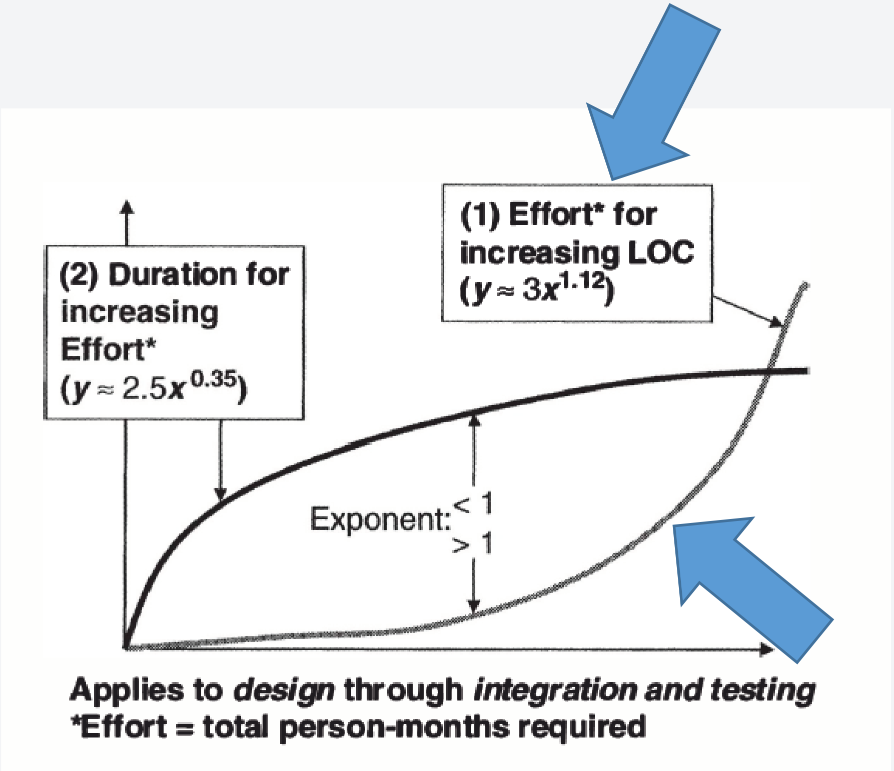
they can be used to estimate effort and project duration using Barry Boehm's COCOMO models.

- COCOMO is based on the idea that **project outcomes, plotted as graphs, have a consistent basic shape.**
- A parameterized formula is found for the shape, so that to obtain the graph for a particular project, he simply has to determine the parameters for it.

Source: Software Engineering Modern Approache E. J. Braude, M. E. Bernstein 2016

COCOMO I

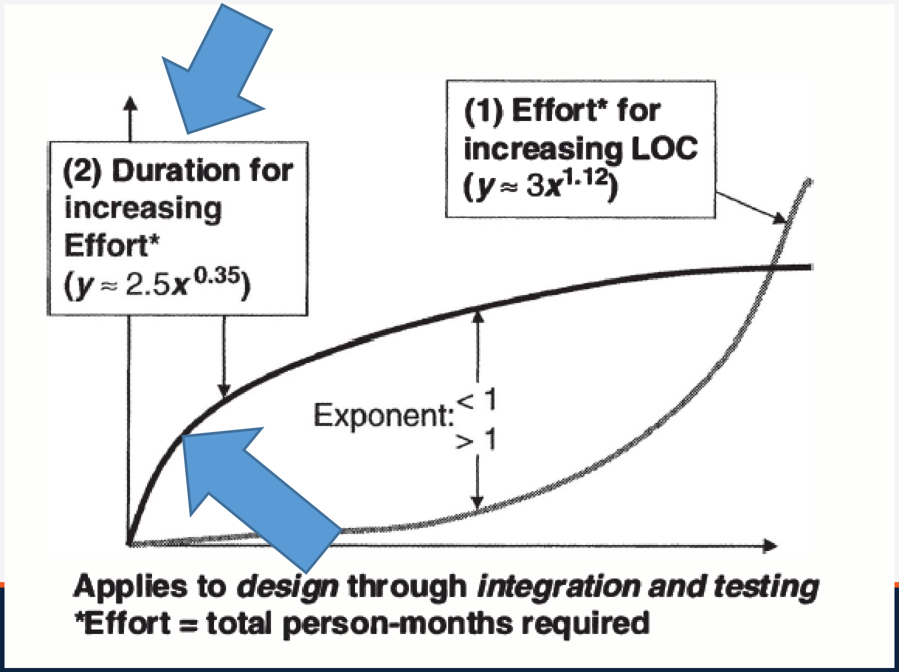
- Boehm observed that the **labor required to develop applications tends to increase faster than the application's size.**
  - Boehm found in his initial research that the exponential function, with exponent close to `1.12`, expresses this relationship quite well.



Source: Software Engineering Modern Approache E. J. Braude, M. E. Bernstein 2016

COCOMO I

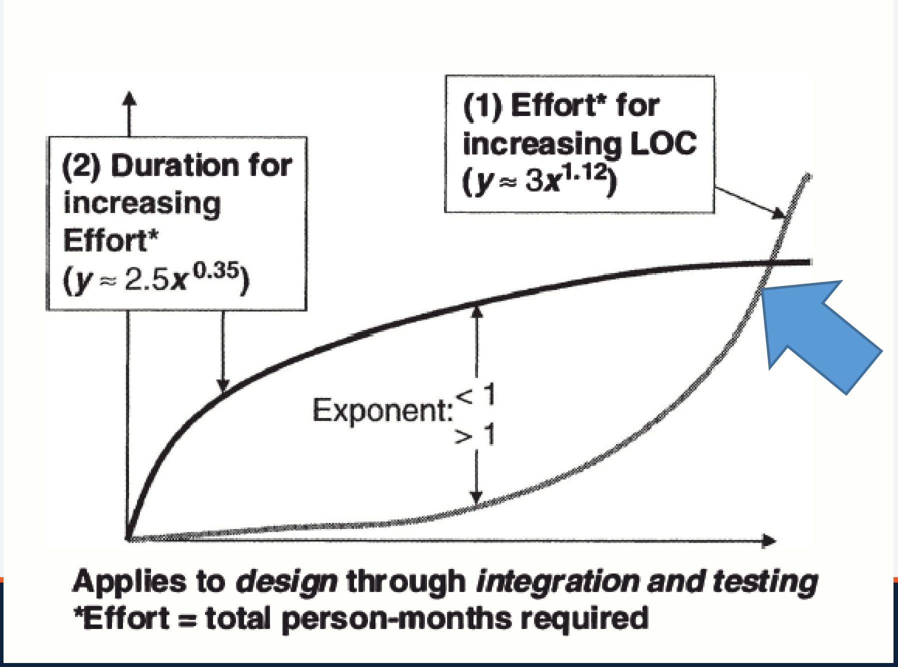
- Boehm's model also says that **the duration increases exponentially with the effort, but with an exponent less than 1** (the exponent used in this case is close to `0.35`).



Source: Software Engineering Modern Approache E. J. Braude, M. E. Bernstein 2016

COCOMO I

- **After a certain size (the "knee" in curve (2)), additional required effort has only a gradual lengthening effect on the time** it takes to complete the project.



Source: Software Engineering Modern Approache E. J. Braude, M. E. Bernstein 2016

COCOMO I – basic level

- COCOMO I (a.k.a., COCOMO'81)
  - The underlying software lifecycle is a **waterfall lifecycle**.
  - Ratings + the cost drivers
- Boehm proposed 3 levels of the model: basic, intermediate, detailed.
  - The basic COCOMO'81 model
    - a single-valued, static model that computes software development **effort (and cost)** as a function of **program size** expressed in estimated thousand delivered source instructions (KDSI).

Effort (KDSI)

Source: COCOMO (Constructive Cost Model) Seminar on Software Cost Estimation, presented by Nancy Merlo – Schett

COCOMO I – intermediate level

- COCOMO I (a.k.a., COCOMO'81)
  - The underlying software lifecycle is a **waterfall lifecycle**.
  - Ratings + the cost drivers
- Boehm proposed 3 levels of the model: basic, intermediate, detailed.
  - The intermediate COCOMO'81 model
    - computes software development **effort** as a function of **program size** and **a set of 15 "cost drivers"** that include subjective assessments of
      - product
      - hardware
      - personnel
      - project attributes

Effort (KDSI, Cost Drivers)

Source: COCOMO (Constructive Cost Model) Seminar on Software Cost Estimation, presented by Nancy Merlo – Schett

COCOMO I – advanced level

- COCOMO I (a.k.a., COCOMO'81)
  - The underlying software lifecycle is a **waterfall lifecycle**.
  - Ratings + the cost drivers
- Boehm proposed 3 levels of the model: basic, intermediate, detailed.
  - The advanced or detailed COCOMO'81 model
    - incorporates all characteristics of the intermediate version with an assessment of the cost driver's impact on **each step (analysis, design, etc.)** of the software engineering process.

$$\mathrm{Effort}_{analysis}\ (\mathrm{KDSI},\ \mathrm{Cost\ Drivers})$$

$$\mathrm{Effort}_{design}\ (\mathrm{KDSI},\ \mathrm{Cost\ Drivers})$$

Source: COCOMO (Constructive Cost Model) Seminar on Software Cost Estimation, presented by Nancy Merlo – Schett

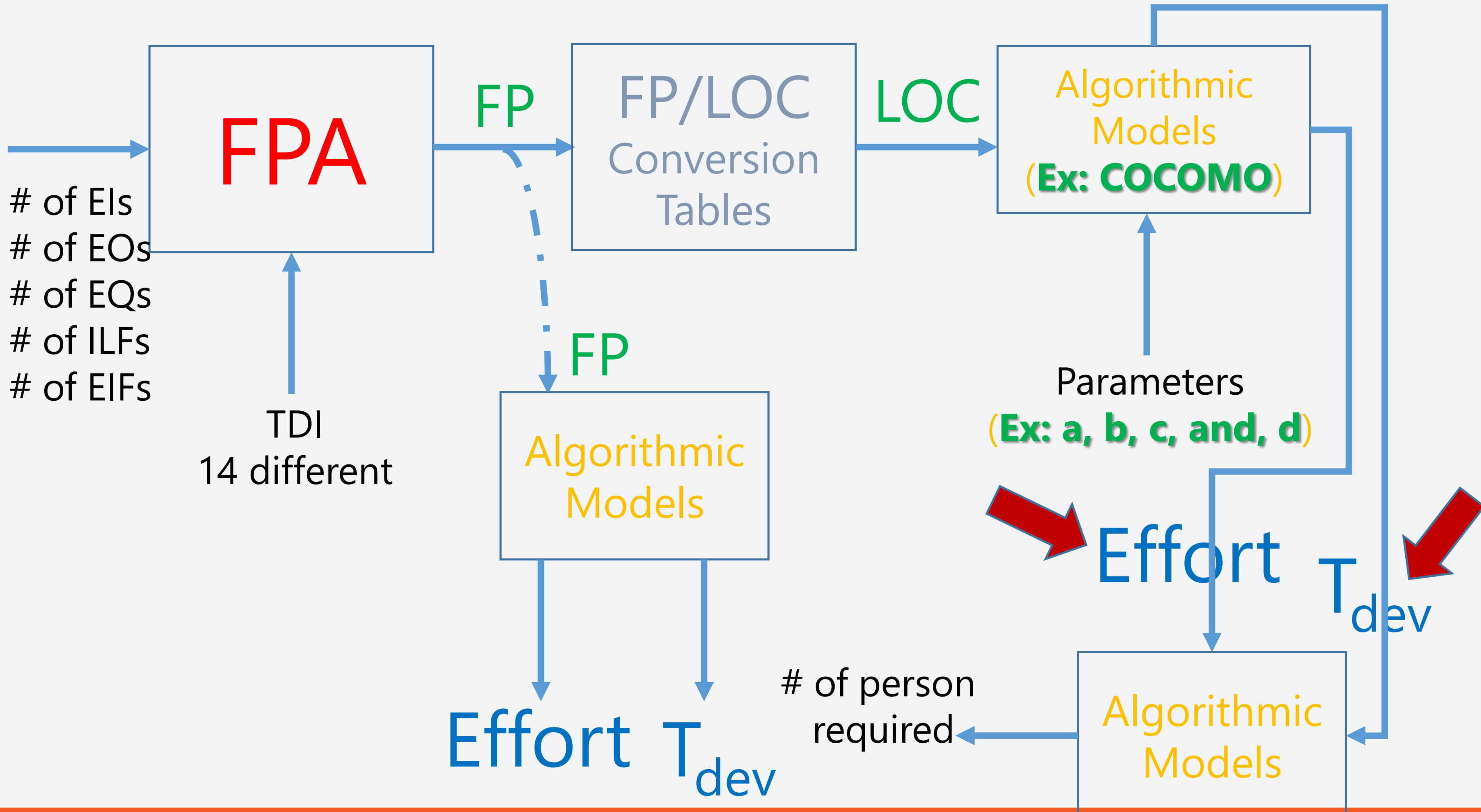
COCOMO I

- COCOMO I depends on the **2 main equations**

1. *Development effort* in *MM = a X KDSI* ^{b}
   - based on MM - man-month or person-month or staff-month is one month of effort by one person.
   - In COCOMO'81, there are **152 hours** per Person-month.
   - According to organization this values may differ from the standard by 10% - 20%.
2. *Effort and development time* (T_{DEV}) : $T_{DEV} = 2.5 \times MM^{c}$
   - The coefficients *a*, *b* and *c* depend on the ***mode of the development***.
   - There are 3 modes of development:
     1. Organic
     2. Semi-detached
     3. Embedded

152 hours per Person-month=>
1 Person-month = 19 Person-day of 8 hours each

Source: COCOMO (Constructive Cost Model) Seminar on Software Cost Estimation, presented by Nancy Merlo – Schett


FPA
# of EIs
# of EOs
# of EQs
# of ILFs
# of EIFs
TDI
14 different
FP
FP/LOC
Conversion
Tables
LOC
Algorithmic
Models
(Ex: COCOMO)
Parameters
(Ex: a, b, c, and, d)
FP
Algorithmic
Models
Effort
T
dev
Effort
T
dev
# of person
required
Algorithmic
Models

COCOMO I - Modes of Development

Development Mode	Project Characteristics			
	Size	Innovation	Deadline/constraints	Dev. Environment
Organic	Small	Little	Not tight	Stable
Semi-detached	Medium	Medium	Medium	Medium
Embedded	Large	Greater	Tight	Complex hardware/ customer interfaces

Source: COCOMO (Constructive Cost Model) Seminar on Software Cost Estimation, presented by Nancy Merlo – Schett

COCOMO I – Basic level

- The basic COCOMO I applies the parameterized equation without much detailed consideration of project characteristics.

1. *Development effort* in $MM = a \times KDSI^{b}$
2. *Effort and development time* (T_{DEV}) : $T_{DEV} = 2.5 \times MM^{c}$

Basic COCOMO	a	b	c
Organic	2.4	1.05	0.38
Semi-detached	3.0	1.12	0.35
Embedded	3.6	1.20	0.32

Source: COCOMO (Constructive Cost Model) Seminar on Software Cost Estimation, presented by Nancy Merlo – Schett

COCOMO I – Intermediate Level

- The same basic equation for the model is used, but 15 cost drivers are rated on a scale of 'very low' to 'very high' to calculate the specific effort multiplier and each of them returns an adjustment factor which multiplied yields in the total EAF (Effort Adjustment Factor).
  - The adjustment factor is 1 for a cost driver that's judged as normal.

Source: COCOMO (Constructive Cost Model) Seminar on Software Cost Estimation, presented by Nancy Merlo – Schett

COCOMO I – Intermediate Level

1. *Development effort* in $MM_{nominal} = a \times KDSI^{b}$

- MM_{corr} = ***EAF*** $\times MM_{nominal}$

2. *Effort and development time* (T_{DEV}) : $TDEV = 2.5 \times MM^{c}$
   - The model parameter "*a*" is slightly different in Intermediate COCOMO from the basic model.
   - The parameter "*b*" remains the same in both models.

Basic COCOMO	a	b	c
Organic	2.4	1.05	0.38
Semi-detached	3.0	1.12	0.35
Embedded	3.6	1.20	0.32

Intermediate COCOMO	a	b	c
Organic	3.2	1.05	0.38
Semi-detached	3.0	1.12	0.35
Embedded	2.8	1.20	0.32

COCOMO I – Advanced Level

- The Advanced COCOMO model computes **effort** as a function of program **size** + **a set of cost drivers** weighted according to **each phase** of the software lifecycle.
  - The Advanced model applies the Intermediate model at the component level, and then a phase-based approach is used to consolidate the estimate.
  - The 4 phases used in the detailed COCOMO model are:
    - Requirements planning and product design (RPD)
    - Detailed design (DD)
    - Code and unit test (CUT)
    - Integration and test (IT)

Source: COCOMO (Constructive Cost Model) Seminar on Software Cost Estimation, presented by Nancy Merlo – Schett

COCOMO I – Advanced Level

- Estimates for each module are combined into subsystems and eventually an overall project estimate.
  - Using the detailed cost drivers, an estimate is determined for each phase of the lifecycle.

Source: COCOMO (Constructive Cost Model) Seminar on Software Cost Estimation, presented by Nancy Merlo – Schett

COCOMO I – Advanced Level

- Each cost driver is broken down by phases as in the example shown in the table below.

Cost Driver	Rating	RPD	DD	CUT	IT
ACAP	Very Low	1.80	1.35	1.35	1.50
	Low	0.85	0.85	0.85	1.20
	Nominal	1.00	1.00	1.00	1.00
	High	0.75	0.90	0.90	0.85
	Very High	0.55	0.75	0.75	0.70

Analyst **CAP**ability effort multiplier for detailed COCOMO

Source: COCOMO (Constructive Cost Model) Seminar on Software Cost Estimation, presented by Nancy Merlo – Schett

COCOMO I – Intermediate - Example

- Required: Database system for an **office automation project**.
- Project → organic (a=3.2, b = 1.05, c=0.38),
- 4 modules to implement
  - data entry → 0.6 KDSI
  - data update →0.6 KDSI
  - query → 0.8 KDSI
  - report generator → 1.0 KDSI
  - System SIZE → 3.0 KDSI

Intermediate COCOMO	a	b	c
Organic	3.2	1.05	0.38
Semi-detached	3.0	1.12	0.35
Embedded	2.8	1.20	0.32

Source: COCOMO (Constructive Cost Model) Seminar on Software Cost Estimation, presented by Nancy Merlo – Schett

COCOMO I – Intermediate - Example

Development effort in *MM = a X KDSI* ^{b}

- MM_{corr} = **EAF** $X\ MM_{nominal}$
- MM_{corr} = (1.15*1.06*1.13*1.17) X 3.2 X 3.0 $^{1.05}$
- MM_{corr} = 16.33

Intermediate COCOMO	a	b	c
Organic	3.2	1.05	0.38
Semi-detached	3.0	1.12	0.35
Embedded	2.8	1.20	0.32

Efforts are rated as follows (all others nominal, 1.0):

cost drivers	level	EAF
complexity	high	1.15
storage	high	1.06
experience	low	1.13
prog capabilities	low	1.17

Source: COCOMO (Constructive Cost Model) Seminar on Software Cost Estimation, presented by Nancy Merlo – Schett

COCOMO I – Intermediate - Example

- **Cost Drivers**
- Intermediate COCOMO introduces Cost Drivers
- They are used because
  - they are statistically significant to the cost of the project
  - they are NOT correlated to the project size (KLOC)

COCOMO I – Intermediate - Example

- **Categories of Cost Drivers**
  1. Product Attributes
  2. Computer Attributes
  3. Personnel Attributes
  4. Project Attributes

COCOMO I – Intermediate - Example

- **Categories of Cost Drivers**
  1. Product Attributes (3)
     - RELY Required Software Reliability
     - DATA Data Base Size
     - CPLX Product Complexity
  2. Computer Attributes
  3. Personnel Attributes
  4. Project Attributes

Efforts are rated as follows (all others nominal, 1.0):

cost drivers	level	EAF
complexity	high	1.15
storage	high	1.06
experience	low	1.13
prog capabilities	low	1.17

COCOMO I – Intermediate - Example

- **Categories of Cost Drivers**

1. Product Attributes
2. Computer Attributes (4)
   - TIME Execution Time Constraint
   - STOR Main Storage Constraint
   - VIRT Virtual Machine Volatility
   - TURN Computer Turnaround Time
3. Personnel Attributes
4. Project Attributes

Efforts are rated as follows (all others nominal, 1.0):

cost drivers	level	EAF
complexity	high	1.15
storage	high	1.06
experience	low	1.13
prog capabilities	low	1.17

COCOMO I – Intermediate - Example

- **Categories of Cost Drivers**
  1. Product Attributes
  2. Computer Attributes
  3. Personnel Attributes (5)
     - ACAP Analyst Capability
     - AEXP Application Experience
     - PCAP Programming Capability
     - VEXP Virtual Machine Experience
     - EXP Programming Language Experience
  4. Project Attributes

Efforts are rated as follows (all others nominal, 1.0):

cost drivers	level	EAF
complexity	high	1.15
storage	high	1.06
experience	low	1.13
prog capabilities	low	1.17

COCOMO I – Intermediate - Example

- **Categories of Cost Drivers**
  1. Product Attributes
  2. Computer Attributes
  3. Personnel Attributes
  4. Project Attributes (3)
     - MODP Modern Programming Practices
     - TOOL Use of Software Tools
     - SCED Required Development Schedule

COCOMO I – Intermediate - Example

- **Cost Drivers**
- Intermediate COCOMO introduces Cost Drivers
- Cost Drivers are used because
  - they are statistically significant to the cost of the project
  - they are NOT correlated to the project size (KLOC)

COCOMO I – Intermediate - Example

Development effort in *MM = a X KDSI* ^{b}

- MM_{corr} = 16.33

- *Effort & dev time* (T_{DEV}) : T_{DEV} = *2.5 X MM* ^{c}
- T_{DEV} = *2.5 X* 16.33 $^{0.38}$
- T_{DEV} = 7.23 (>7months to complete)

Intermediate COCOMO	a	b	c
Organic	3.2	1.05	0.38
Semi-detached	3.0	1.12	0.35
Embedded	2.8	1.20	0.32

COCOMO I – Intermediate - Example

Development effort in *MM* = $a \, X \, KDSI^{\,b}$

- $MM_{corr} = 16.33$

PM: I have already 16 SWEs in my team. Can we complete the project in 1 months?

Source: COCOMO (Constructive Cost Model) Seminar on Software Cost Estimation, presented by Nancy Merlo – Schett

COCOMO I – Intermediate - Example

- How many people should be hired?
  - MM_{corr} / TDEV = team members
  - 16.33 / 7.23 = 2.26 (>2 team members)

Source: COCOMO (Constructive Cost Model) Seminar on Software Cost Estimation, presented by Nancy Merlo – Schett

COCOMO I – Intermediate

- **Categories of Cost Drivers**
    1. Product Attributes
    2. Computer Attributes
    3. Personnel Attributes
    4. Project Attributes

	Description	*Very Low*	*Low*	*Nominal*	*High*	*Very High*	*Extra High*
RELY	Required software reliability	0.75	0.88	1.00	1.15	1.40	-
DATA	Database size	-	0.94	1.00	1.08	1.16	-
CPLX	Product complexity	0.70	0.85	1.00	1.15	1.30	1.65

Source: https://people.ucalgary.ca/~far/Lectures/SENG421/PDF/SENG421-07.pdf

COCOMO I – Intermediate

- **Categories of Cost Drivers**
  1. Product Attributes
  2. Computer (Platform) Attributes
  3. Personnel Attributes
  4. Project Attributes

	Description	*Very Low*	*Low*	*Nominal*	*High*	*Very High*	*Extra High*
TIME	Execution time constraint	-	-	1.00	1.11	1.30	1.66
STOR	Main storage constraint	-	-	1.00	1.06	1.21	1.56
VIRT	Virtual machine volatility	-	0.87	1.00	1.15	1.30	-
TURN	Computer turnaround time	-	0.87	1.00	1.07	1.15	-

COCOMO I – Intermediate

- **Categories of Cost Drivers**
  1. Product Attributes
  2. Computer Attributes
  3. Personnel Attributes
  4. Project Attributes

	Description	*Very Low*	*Low*	*Nominal*	*High*	*Very High*	*Extra High*
ACAP	Analyst capability	1.46	1.19	1.00	0.86	0.71	-
AEXP	Applications experience	1.29	1.13	1.00	0.91	0.82	-
PCAP	Programmer capability	1.42	1.17	1.00	0.86	0.70	-
VEXP	Virtual machine experience	1.21	1.10	1.00	0.90	-	-
LEXP	Language experience	1.14	1.07	1.00	0.95	-	-

COCOMO I – Intermediate

- **Categories of Cost Drivers**
  1. Product Attributes
  2. Computer Attributes
  3. Personnel Attributes
  4. Project Attributes

	Description	*Very Low*	*Low*	*Nominal*	*High*	*Very High*	*Extra High*
MODP	Modern programming practices	1.24	1.10	1.00	0.91	0.82	-
TOOL	Software Tools	1.24	1.10	1.00	0.91	0.83	-
SCED	Development Schedule	1.23	1.08	1.00	1.04	1.10	-

COCOMO I – Intermediate Cost Driver Rating: Example

- **Example:** Required software Reliability **(RELY)**
- Measures the extent to which the software must perform its intended function over a period of time.
- **Ask:** What is the *effect of a software failure*?

	Very Low	Low	Nominal	High	Very High	Extra High
RELY	slight inconvenience	low, easily recoverable losses	moderate, easily recoverable losses	high financial loss	risk to human life	
	0.75	0.88	1.00	1.15	1.39	

Source: https://people.ucalgary.ca/~far/Lectures/SENG421/PDF/SENG421-07.pdf

COCOMO I - Advantages

- COCOMO is transparent, one can see how it works unlike other models such as SLIM
- Drivers are particularly helpful to the estimator to understand the impact of different factors that affect project costs

Source: COCOMO (Constructive Cost Model) Seminar on Software Cost Estimation, presented by Nancy Merlo – Schett

COCOMO I - Disadvantages

- It is hard to accurately estimate KDSI early in the project, when most effort estimates are required KDSI, actually, is not a size measure it is a length measure
- Extremely vulnerable to mis-classification of the development mode
- Success depends largely on tuning the model to the needs of the organization, using historical data which is not always available

Source: COCOMO (Constructive Cost Model) Seminar on Software Cost Estimation, presented by Nancy Merlo – Schett

COCOMO I – Wrap-up

COCOMO I

- Boehm's model says first that **the required effort and duration have separate models (formulas) for each type of application** (differing in factors `a` and `b`) .
  - **Ex 1:** A stand-alone job with 20,000 LOC would take $2.4 \times 20^{1.05} \approx 56$ PM duration if organic (stand-alone).
  - **Ex 2:** But the same job would take $3.6 \times 20^{1.2} \approx 131$ person-months if embedded.
- The duration formula can be expressed directly in terms of KLOC as follows:
- Duration = $c \times Effort^{d} = c \times (a \times KLOC^{b})^{d} = c \times a^{d} \times KLOC^{bd}$

$$\text{Effort in Person-months} = a \times KLOC^{b}$$
$$\text{Duration} = c \times Effort^{d}$$

Software Project	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semi-detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Source: Software Engineering Modern Approache E. J. Braude, M. E. Bernstein 2016

COCOMO I

- At first glance, Boehm's duration formula may appear strange because the relationship between effort and duration seems to be simpler than his formula indicates.
- For example, `if we know that a job requires 120 PM, and we put 10 person onto it, won't it get done in 12 months?`
  - This would indeed be the case if we could **usefully** and **consistently** employ all 10 people on the project from day 1 through day 365, but this is NOT usually possible.
    - Consider, for example, day 1. Since all SWEs can't know much about the project (it has just begun), **what useful activities could ALL 10 engineers do on day 1?**
  - It follows that if we allocate 10 SWEs from the day 1, the 120 PM job will actually take longer than 12 months.

Source: Software Engineering Modern Approache E. J. Braude, M. E. Bernstein 2016

COCOMO I

- Boehm's duration formula has the strange property of **being independent of the # of people put on the job**!
- **It depends only on the size of the job.**
- Actually, the formula assumes that the project will have **roughly appropriate # of people** available to it at any given time.
  - **Ex:** 1 person on day 1, 30 persons on day 100, assuming that is what's needed.

Source: Software Engineering Modern Approache E. J. Braude, M. E. Bernstein 2016

$$\text{Effort in Person-months} = a \times KLOC^{b}$$
$$\text{Duration} = c \times Effort^{d}$$

COCOMO I

- Using Boehm's basic formula on a sample project, with 4- 300 KLOC, we obtain
  - 10 to 1,000 PM of effort
  - 6 to 35 months in duration

		a	K	b		approx.
Effort						aK^b
	LO	2.4	4.2	1.05		10
	HI	2.4	300	1.05		1000

		c	P	d		approx.
Duration						cP^d
	LO	2.5	10	0.38		6
	HI	2.5	1000	0.38		35

Source: Software Engineering Modern Approache E. J. Braude, M. E. Bernstein 2016

$$\text{Effort in Person-months} = a \times KLOC^b$$
$$\text{Duration} = c \times Effort^d$$

COCOMO I

- Using Boehm's basic formula on a sample project, with 4- 300 KLOC, we obtain
  - 10 to 1,000 PM of effort
  - 6 to 35 months in duration

		a	K	b		approx.
Effort						aK^b
	LO	2.4	4.2	1.05		10
	HI	2.4	300	1.05		1000

		c	P	d		approx.
Duration						cP^d
	LO	2.5	10	0.38		6
	HI	2.5	1000	0.38		35