High performance. Delivered.

TU München
Lecture Applied Software Engineering
Prof. Bernd Brügge
Software Engineering II, SS 2009

Guest Lecture: Estimation
Wolfgang Behr, Accenture

Lecture Schedule (Less Tentative)

- Apr 21: **Introduction**
- Apr 28: **Basic Concepts**
- May 5: **Project Communication**
- May 13: **Configuration Management**
- May 19: **Build and Release Management**
- May 26: **Estimation** (Guest Speaker: Wolfgang Behr, Accenture)
- June 02: Cancelled (Pentecost, Pfingsten)
- June 09: **Scheduling** (Project duration, critical path analysis)
- June 16: Guest lecture (Holger Wolff, Beck et al)
- June 23: **Organization** (team building, customer interaction)
- June 30: **Lifecycle Models** (Lifecycle models, IEEE 1074, Unified Process)
- July 7: **Agile Project Management** (XP, Scrum)
- July 14 : Corporate Open Source (Guest Lecture, Thomas Uhl, Topalis AG)
- July 21: **Knowledge Management** (Acquiring and externalizing knowledge, dealing with conflicts and resolutions)
- July 30: Exam (14:30 – 16:30 in MW 1350)

Exercise Schedule (Less Tentative)

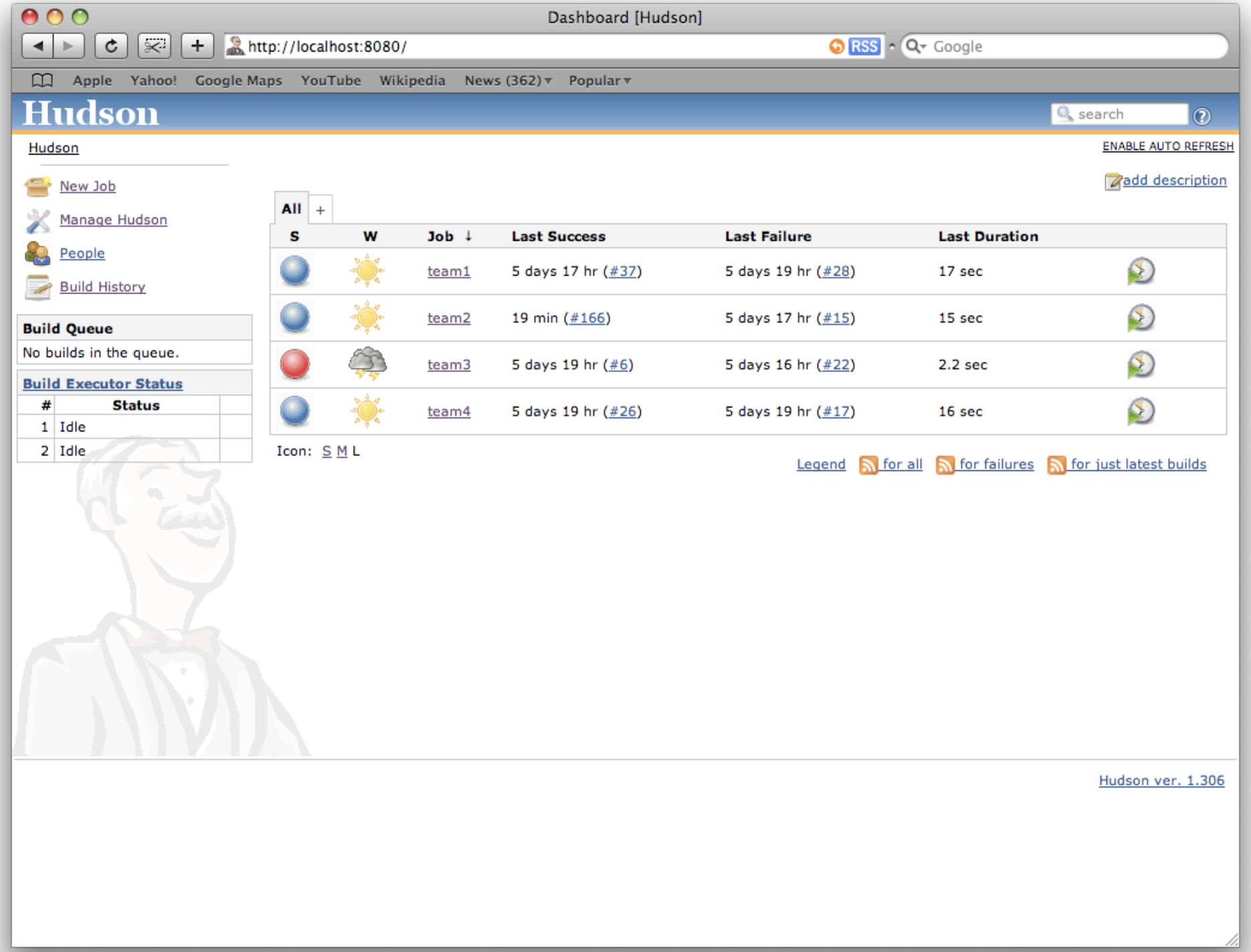
- April 22: Icebreaker
- April 29: Software Project Management Plan (SPMP)
- May 6: Project Agreement
- May 13: Software Configuration Management Plan (SCMP)
- May 20: Continuous Integration (Cruise Control, Maven)
- Fr. May 29: Work Breakdown structures (Jonas von Beck, Accenture)
  - Preparation: Read WBS lecture slides, Slides are posted on Lecture Portal
- Fr. June 5: Estimation (Marc Bachmann, Accenture)
- June 10: Scheduling (Inga Küffer, Accenture)
- **Week of June 15-20: Project Management Day at Accenture (Block Event) Exact date to be announced**
- June 24: Rationale Management
- July 1: Student presentations of SPMP
- July 8: Agile Project Management (Daily Scrum, Planning Poker)

Accenture Schedule

- Starting today with Wolfgang Behr's lecture on WBS
- The following 4 exercises will be held in cooperation with Accenture.
  - Friday May 29: Work Breakdown structures (Create a WBS)
  - Friday June 5: Estimation (Establish Estimates)
  - Wednesday June 10: Scheduling (Set up a project schedule)
  - Week of June 15-20:
    - Project Management Day at Accenture
      (Block Event) Exact date to be announced
- Exercises on Fridays take place from 2 - 3:30 pm in room 01.07.014

Continuous Integration Exercise Post-Mortem

- What went right?
  - 27 highly motivated students in 4 teams
  - All team were able to set up the Hudson project correctly
  - Great communication and teamwork


Dashboard [Hudson]
http://localhost:8080/
RSS
Google
Apple
Yahoo!
Google Maps
YouTube
Wikipedia
News (362)
Popular
Hudson
search
Hudson
ENABLE AUTO REFRESH
New Job
Manage Hudson
People
Build History
add description
Build Queue
No builds in the queue.
Build Executor Status
Status
1 Idle
2 Idle
All
+
S W Job ↓ Last Success Last Failure Last Duration
team1 5 days 17 hr (#37) 5 days 19 hr (#28) 17 sec
team2 19 min (#166) 5 days 17 hr (#15) 15 sec
team3 5 days 19 hr (#6) 5 days 16 hr (#22) 2.2 sec
team4 5 days 19 hr (#26) 5 days 19 hr (#17) 16 sec
Icon: S M L
Legend
for all
for failures
for just latest builds
Hudson ver. 1.306

Continuous Integration Exercise Post-Mortem

- What went wrong?
  - Ad hoc network infrastructure (DHCP Problems)
  - Bad planning (teams ran out of time)
  - Tests provided by management were underspecified or even without any specification
    - But, the teams found the problems

Continuous Integration Exercise Post-Mortem

- Who won the ice-cream?
  - All the teams managed well for the short time available
  - One team configured all the metrics plug-ins correctly and managed to eliminate all PMD warnings and at least some of the other warnings.
  - The winner is:

TEAM 4

Objectives for Today

Build an understanding of...

- Importance
- Challenges
- Approaches
- Pros and Cons
- Pitfalls

What is a Project Estimate ?

- A declaration about needed
  - effort and
  - time,
  - for delivering the project scope

Importance of Estimations

- Ressource allocation decisions
- Basis for the decision to start (or not to start) a project
- Foundation for project planning and set-up (business case)
- Foundation for project controlling
- If project time is a given, number of ressources can be determined
- "Owner" of an estimate is an indication about who is taking the project risk
- Decision and ressource allocation implications => Estimates are often part of "political games"

Estimating is a core task of project management

Challenges (1/2)

- Incomplete knowledge about:
  - Project scope and changes
  - Prospective resources and staffing
  - Technical and organizational environment
  - Infrastructure
  - Feasibility of functional requirements

- Comparability of projects in case of new or changing technologies, staff, methodologies
- Learning curve problem
- Different expectations towards project manager

Challenges (2/2)

- Estimation is too low
  - Scope and tasks (WBS) incomplete / unknown
- Estimation is too high
  - Political / human reasons
  - Learning curve
- New technologies can make new parameters necessary

Guiding Principles

- Documentation of assumptions about
  - Estimation methodology
  - Project scope, staffing, technology, ...
- Definition of estimation accuracy
- Continuous planning and estimation over project time (increasing accuracy with project phases)
  - Example: Better estimation for implementation phase after object design is finished
- Reviews by experienced colleagues
- Depending on the situation, multiple methods are to be used in combination

Components of an Estimation

This lecture

- Cost
  - Personnel (in person days or valued in personnel cost)
    - Person day: Effort of one person per working day
  - Material (PCs, software, tools etc.)
  - Extra costs (travel expenses etc.)
- Development Time
  - Project duration
  - Dependencies
- Infrastructure
  - Rooms, technical infrastructure, especially in offshore scenarios.

Lecture on Scheduling.

Estimating Development Time

Development time often estimated by formula

Duration = Effort / People

But:

- A larger project team increases communication complexity which usually reduces productivity

- Therefore it is not possible to reduce duration arbitrarily by adding more people to a project

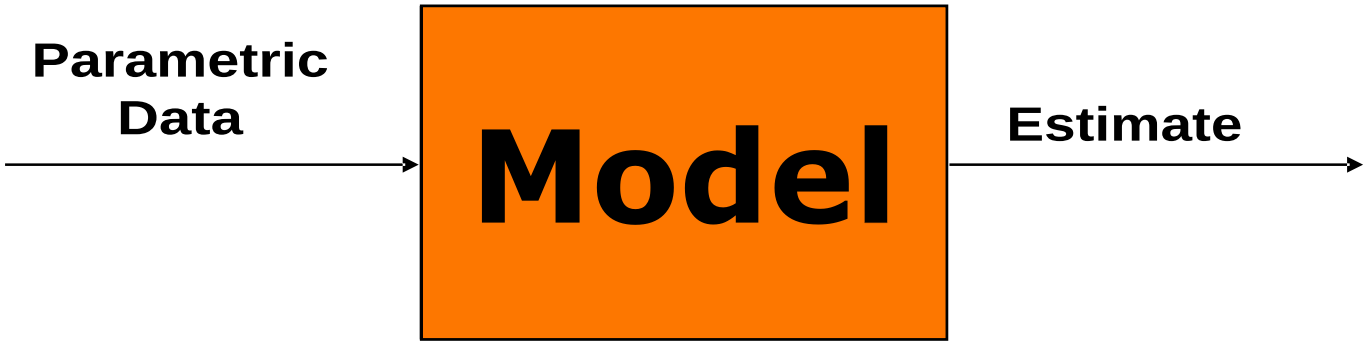
Estimating Personnel Cost

- Staff categories (based on experience, qualification and skills), for example:
  - teamlead, junior business analyst, senior business analyst, junior programmer, senior programmer, subject matter expert
- Cost rate: Cost per person per day
  - 2 alternatives for cost rate:
    - Single cost rate for all types (no differentiation necessary)
    - Assign different cost rates to different categories
- Personnel cost: person days x cost rate.

Estimating Effort

- Most difficult part during project planning
  - Many planning tasks (project schedule, project organization) depend on determination of effort
- Basic principle:
  - Select an estimation model (or build one first)
  - Evaluate known information: project scope, resources, software process (for example documentation requirements), system components
  - Feed this information as parametric input data into the model
  - Model converts the input into an estimate about the effort

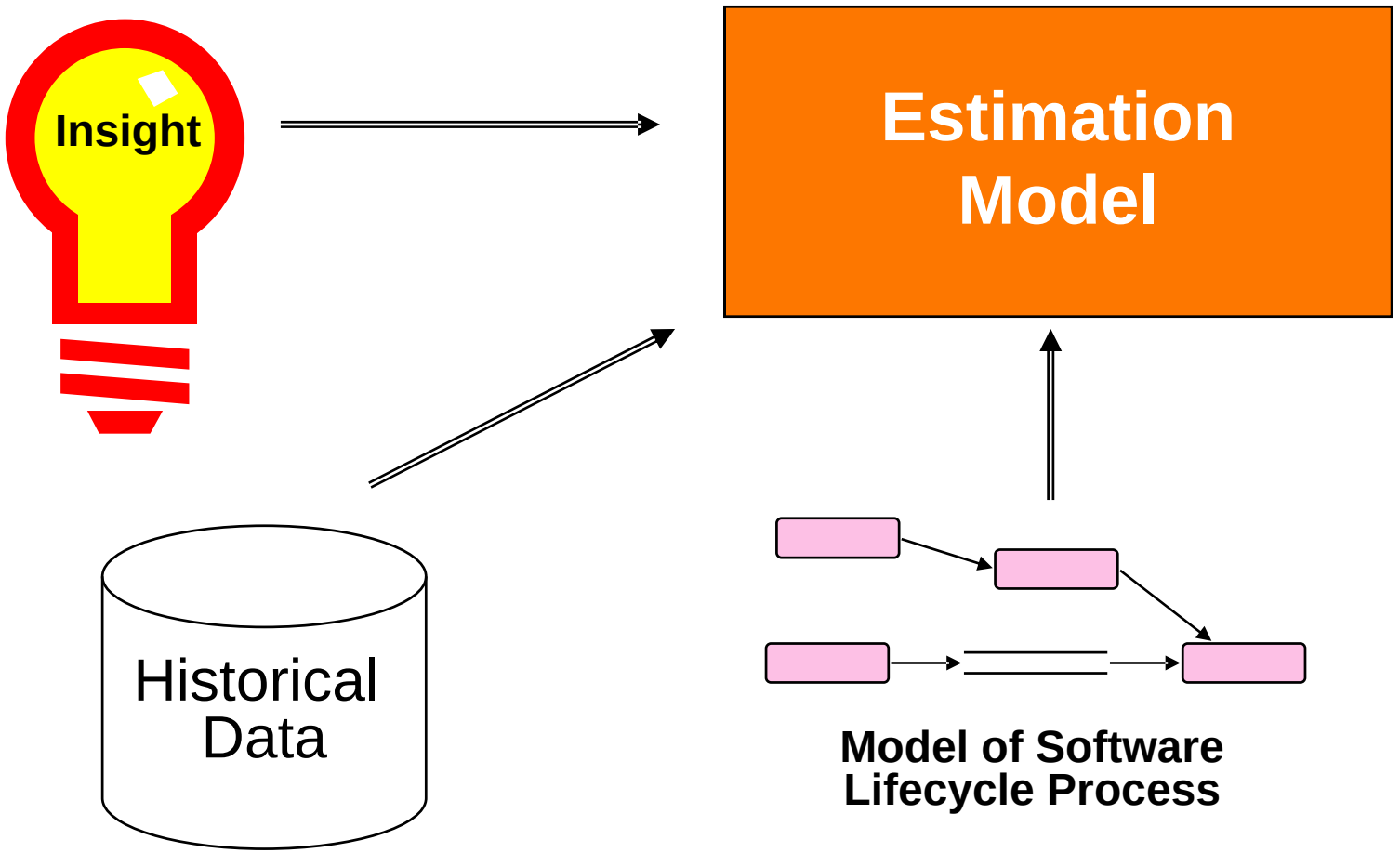
Basic Use of Estimation Models



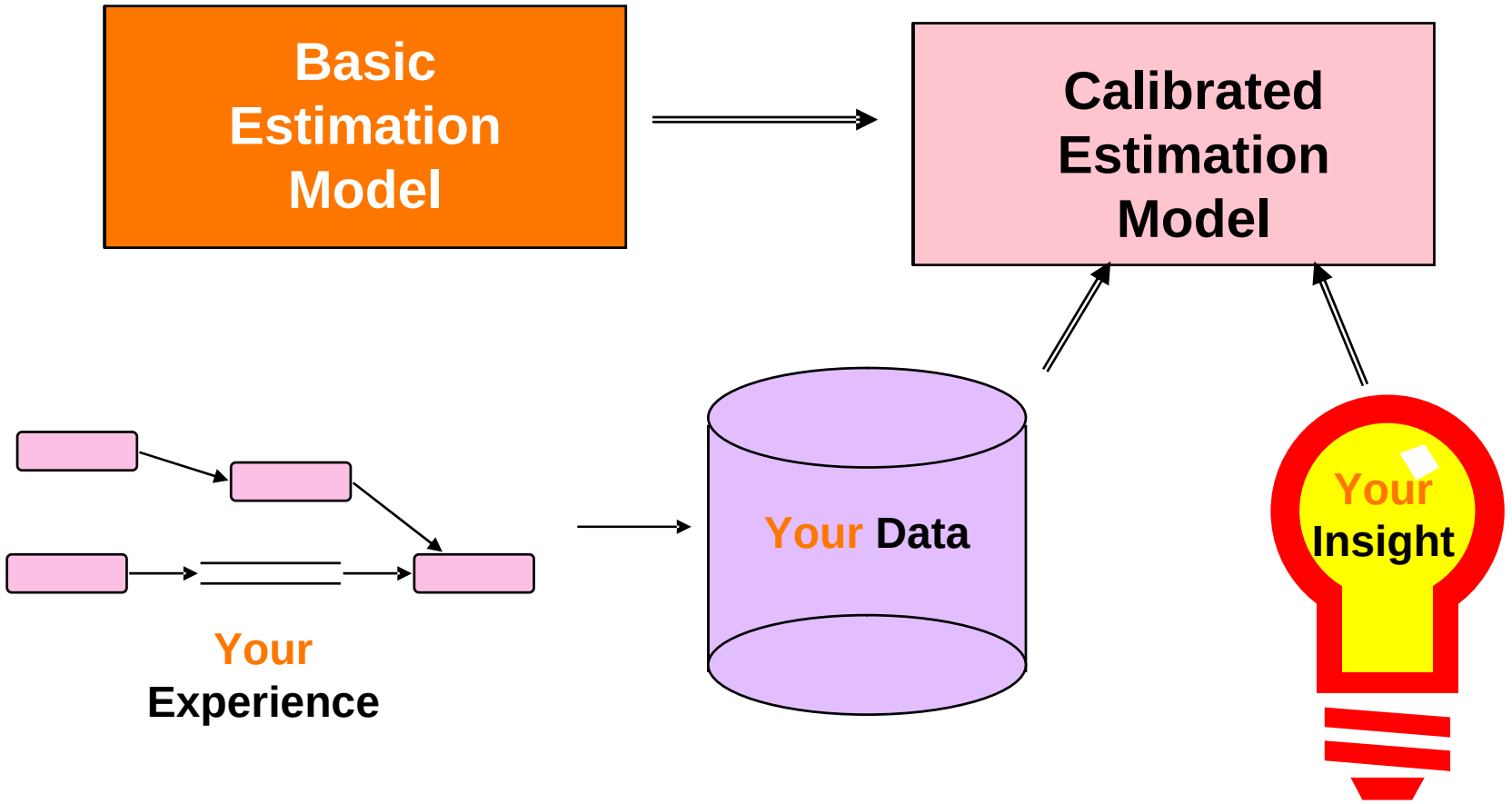
Examples:

Data Input	Estimate
Size & Project Data	Effort & Schedule
System Model	Performance
Software Process	Cycle Time.

How do you Build an Estimating Model?



Calibrating an Estimation Model



Top-Down and Bottom-Up Estimation

- Two common approaches for estimations
  - Top-Down Approach
    - Estimate effort for the whole project
    - Breakdown to different project phases and work products
  - Bottom-Up Approach
    - Start with effort estimates for tasks on the lowest possible level
    - Aggregate the estimates until top activities are reached.

Top-Down versus Bottom-Up (cont'd)

- Top-Down Approach
  - Normally used in the planning phase when little information is available how to solve the problem
  - Based on experiences from similar projects
  - Not appropriate for project controlling (too high-level)
  - Risk add-ons usual as result tends to be too low
- Bottom-Up Approach
  - Normally used after activities are broken down to task level and estimates for the tasks are available
  - Result can be used for project controlling (detailed level)
  - Smaller risk add-ons (tends to be too high)
- Often a mixed approach with recurring estimation cycles is used.

Estimation Techniques

- Expert estimations
- Lines of code
- Function point analysis
- COCOMO
- Estimation Technique used by Accenture.

Expert Estimations

Expert Estimations

= Guess from experienced people

- Mostly used top-down for the whole project, but also for some parts of a bottom-up approach
- Used for determining the calibration parameters
- No better than the participants
- Result justification difficult
- Also suitable for:
  - atypical projects
  - in pre-project / idea phase.

Lines of Code

Lines of Code

- Traditional way for estimating application size (FORTRAN and assembler -> line-oriented languages)
- Advantage: Easy to do
- Disadvantages:
  - No standard definition for "Line of Code" (logical versus physical)
  - Of no help given a written project scope or functional design
  - "You get what you measure": If the number of lines of code is the primary measure of productivity, programmers ignore opportunities of reuse
  - Multi-language environments: Hard to compare mixed language projects with single language projects

"The use of lines of code metrics for productivity should be regarded as professional malpractice" (Caspers Jones).

Function Point Analysis

Function Point Analysis

- Developed by Allen Albrecht, IBM Research, 1979
- Technique to determine size of software projects
  - Size is measured from a functional point of view
  - Estimates are based on functional requirements
- Albrecht originally used the technique to predict effort
  - Size is usually the primary driver of development effort
- Independent of
  - Implementation language and technology
  - Development methodology
  - Capability of the project team
- A top-down approach based on function types
  - Three steps: Plan the count, perform the count, estimate the effort.

Steps in Function Point Analysis

- Plan the count
  - Type of count: development, enhancement, application
  - Identify the counting boundary
  - Identify sources for counting information: software, documentation and/or expert
- Perform the count
  - Count data access functions
  - Count transaction functions
- Estimate the effort
  - Compute the unadjusted function points (UFP)
  - Compute the Value Added Factor (VAF)
  - Compute the adjusted Function Points (FA)
  - Compute the performance factor
  - Calculate the effort in person days.

Function Types

Data function types

- # of internal logical files (ILF)
- # of external interface files (EIF)

Transaction function types

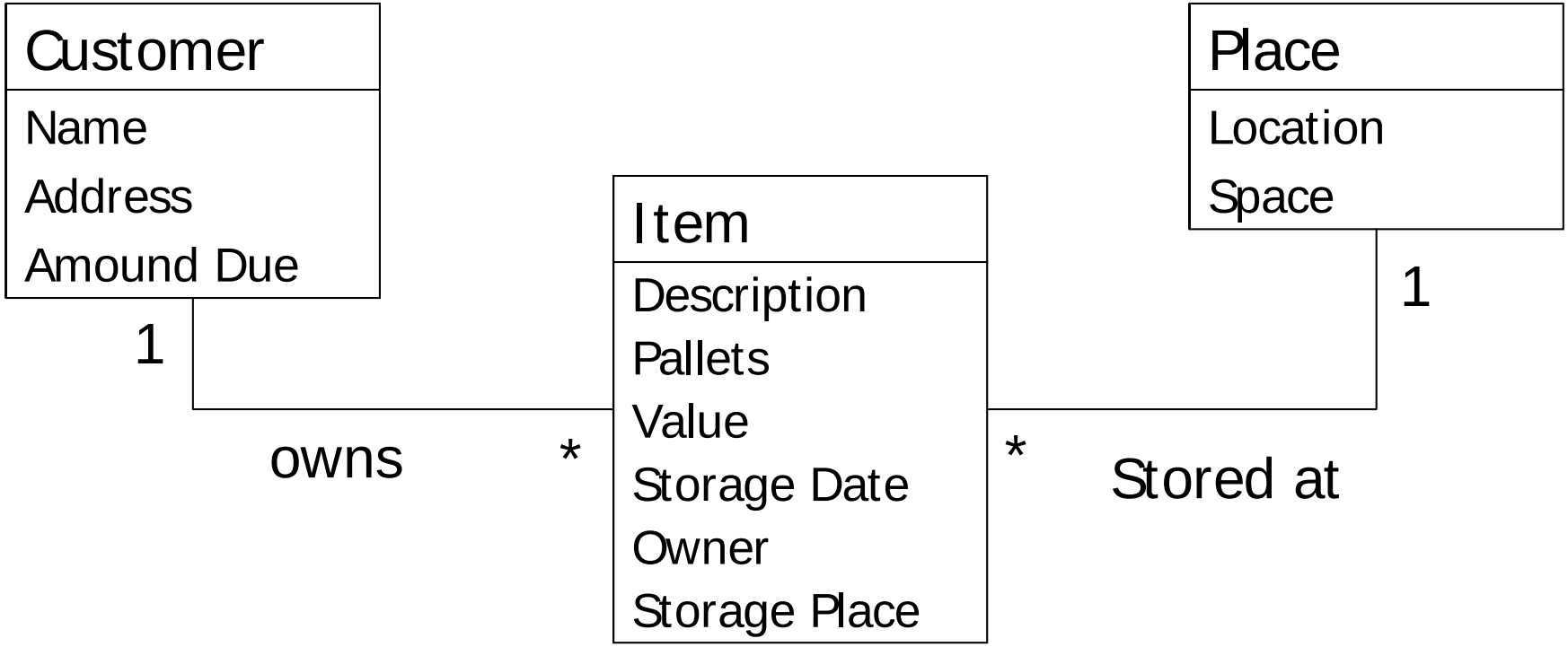
- # of external input (EI)
- # of external output (EO)
- # of external queries (EQ)

Calculate the UFP (unadjusted function points):

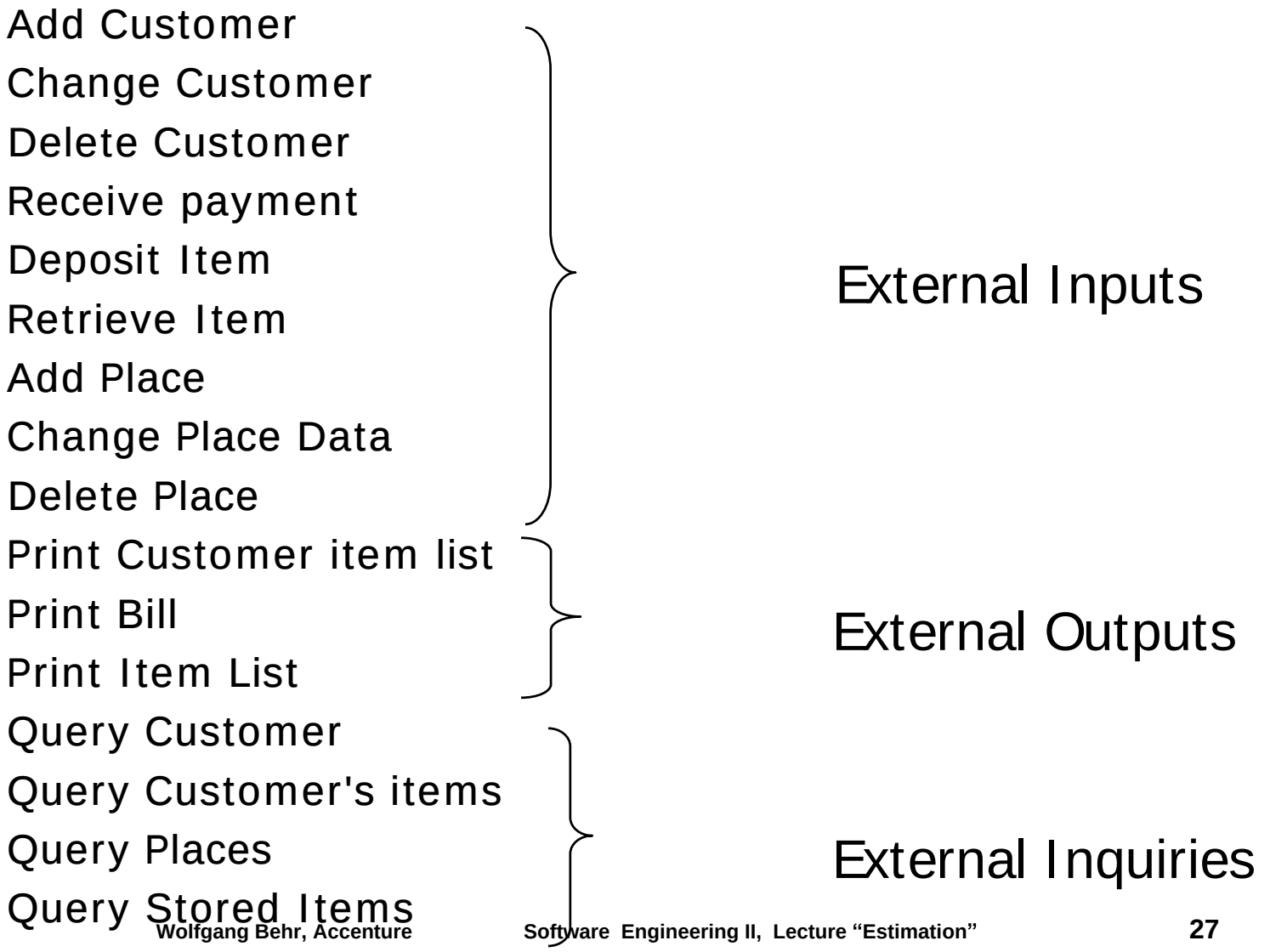
$$UFP = a \cdot EI + b \cdot EO + c \cdot EQ + d \cdot ILF + e \cdot EIF$$

a-f are so-called weight factors (see slide 28)

Object Model Example



Mapping Functions to Transaction Types



Calculate the Unadjusted Function Points

Function Type	Number		Weight Factors: simple	average	complex		
External Input (EI)		x	3	4	6	=	
External Output (EO)		x	4	5	7	=	
External Queries (EQ)		x	3	4	6	=	
Internal Datasets (ILF)		x	7	10	15	=	
Interfaces (EIF)		x	5	7	10	=	
Unadjusted Function Points (UFP)						=	

14 General System Complexity Factors

- The unadjusted function points are adjusted with general system complexity (GSC) factors

GSC1: Reliable Backup & Recovery
GSC2: Use of Data Communication
GSC3: Use of Distributed Computing
GSC4: Performance
GSC5: Realization in heavily used configuration
GSC6: On-line data entry
GSC7: User Friendliness
GSC8: On-line data change
GSC9: Complex user interface
GSC10: Complex procedures
GSC11: Reuse
GSC12: Ease of installation
GSC13: Use at multiple sites
GSC14: Adaptability and flexibility

- Each of the GSC factors gets a value from 0 to 5.

Calculate the Effort

- After the GSC factors are determined, compute the Value Added Factor (VAF):

$$VAF = 0.65 + 0.01 * \sum_{i=1}^{14} GSC_i \qquad GSC_i = 0,1,...,5$$

- Function Points = Unadjusted Function Points * Value Added Factor
  - FP = UFP · VAF
- Performance factor
  - PF = Number of function points that can be completed per day
- Effort = FP / PF

Advantages of Function Point Analysis

- Independent of implementation language and technology
- Estimates are based on design specification
  - Usually known before implementation tasks are known
- Users without technical knowledge can be integrated into the estimation process
  - Incorporation of experiences from different organizations
- Easy to learn
  - Limited time effort.

Disadvantages of Function Point Analysis

- Complete description of functions necessary
  - Often not the case in early project stages -> especially in iterative software processes
- Internal functions (algorithms) rather underestimated, as model is based on user-oriented requirements and functions
- Only complexity of specification is estimated
  - Implementation is often more relevant for estimation
- High uncertainty in calculating function points:
  - Weight factors are usually deducted from past experiences (environment, used technology and tools may be out-of-date in the current project)
- Not suitable for project controlling.

COCOMO

COCOMO (COnstructive COst MOdel)

- Developed by Barry Boehm in 1981
- Also called COCOMO I or Basic COCOMO
- Top-down approach to estimate cost, effort and schedule of software projects, based on size and complexity of projects
- Assumptions:
  - Derivability of effort by comparing finished projects (“COCOMO database”)
  - System requirements do not change during development
  - Exclusion of many efforts (for example administration, training, rollout, integration).

Advantages of COCOMO

- Appropriate for a quick, high-level estimation of project costs
- Fair results with smaller projects in a well known development environment
  - Assumes comparison with past projects is possible
- Covers all development activities (from analysis to testing)
- Intermediate COCOMO yields good results for projects on which the model is based.

Problems with COCOMO

- Model derived from the time when central batch processing was the standard
- Lines of code (software size) needed
- Expert judgment required to determine the influencing factors and their values
- Experience shows that estimation results can deviate from actual effort by a factor of 4!
- Important project factors are not considered:
  - Skills of team members, travel, environmental factors, user interface quality, overhead cost.
- COCOMO 81 (the original model) is out of date, COCOMO II published in 2001

Estimation Technique used by Accenture

Estimation Technique used by Accenture

Uses both top-down and bottom-up elements

Consists of 9 steps:

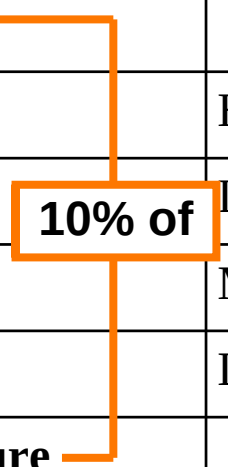
1. Determine essential project characteristics
   - Scope, infrastructure, technology, team skills, experience...
2. Use factors for fixed efforts and phases:
   - Often derived from already finished phases (step-by-step detailling of estimations)
   - Example:
     - 10% for project management
     - 10 % for infrastructure
     - 50% for testing efforts.

Estimation Technique used by Accenture (2)

3. Determine work products for the system to be developed (WBS)
5. Determine work product types (use case, user interface, batch program, ...)
4. Assign a complexity factor to each of these work products
6. Define all necessary activities or tasks that need to be done to produce these work products
7. Assign effort estimates (in person days) to these tasks by using past experience
8. Aggregate the estimates to compute the overall project effort
9. Use add-ons (contingency and risk factors).

Example of Complexity and Multipliers (Non-exhaustive)

	Complexity	Type	Multiplier / Factor	Person Days
Requirements Elicitation				**33**
Function A	Low	Use Case	1	5
Function B	Medium	Use Case	1	8
Function C	High	Use Case	2	20
Implementation				**39**
Screen A	High	User interface	1	18
Screen B	Low	User interface	2	8
Batch Job A	Medium	Batch	1	8
Batch Job B	Low	Batch	1	5
Software Architecture			**10 %**	**3,9**
Sum				**75,9**



Prerequisites for Accenture's Technique

- Identical estimation approach for different projects necessary
- Lots of experience with estimating projects necessary in order to develop good parameters
- Multiple checks of top-down with bottom-up results and vice versa
- Post calculation after end of project important for improving estimation parameters.

Estimation Technique used by Accenture (3)

There are different estimating models available for different situations:

- Top-Down Model (initial estimate for early project phases)
- Bottom-Up Model (detailed estimating model)
- Custom development
- Packaged development (implementation of application software packages like SAP, Siebel, PeopleSoft, Oracle and any other packages)
- Distributed work (using off-shoring for example)
- ...

Summary (1/2)

- Estimation is often the basis for the decision to start, plan and manage a project
- Estimating software projects is a complex project management function
- All approaches depend very much on personal experiences
- If used properly, estimates can be a transparent way to discuss project effort and scope
- However,
  - Few organizations have established formal estimation processes
  - Existing estimation techniques have lots of possibilities to influence the results - must be used with care.

Summary (2/2)

- Even more important than estimating the effort and costs of a development effort is the estimation of the benefits (business case)
  - Example: large German bank
  - Example: German consumer credit bank
- 5-10% estimation variance is usual in a more sophisticated organization
- Methods closer to "agile planning and estimation" techniques are becoming more prevalent, for example by planning quick wins ("sprints"), small releases etc.

Further Readings

- B. Boehm, Software Engineering Economics, Prentice-Hall, 1981
- B. Boehm, Software Cost Estimation With COCOMO II, Prentice Hall, 2000
- D. Garmus, D. Herron, Function Point Analysis: Measurement Practices for Successful Software Projects, Addison-Wesley, 2000
- International Function Point Users Group
  - http://www.ifpug.org/publications/case.htm
- C. Jones, Estimating Software Costs, 1998
- S. Whitemire, Object-Oriented Design Measurement, John Wiley, 1997

Online Availability of Estimation Tools

- Basic and Intermediate COCOMO I (JavaScript)
  - http://www1.jsc.nasa.gov/bu2/COCOMO.html
  - http://ivs.cs.uni-magdeburg.de/sw-eng/us/java/COCOMO/index.shtml
- COCOMO II (Unix, Windows and Java)
  - http://sunset.usc.edu/available_tools/index.html
- Function Point Calculator (Java)
  - http://ivs.cs.uni-magdeburg.de/sw-eng/us/java/fp/

GSC Factors in Function Point Analysis

1. Data communications: How many communication facilities aid in the transfer or exchange of information with the system?☐
2. Distributed data processing:How are distributed data and processing functions handled?☐
3. Performance: Does the user require a specific response time or throughput?☐
4. Platform usage: How heavily used is the platform where the application will run?☐
5. Transaction rate: How frequently are transactions executed (daily, weekly, monthly)?
6. On-line data entry: What percentage of the information is entered On-Line?☐
7. End-user efficiency: Is the application designed for end-user efficiency?☐

GSC Factors in Function Point Analysis (cont'd)

8. On-line update: How many ILF's are updated on-line?
9. Complex processing: Does the application have extensive logical or mathematical processing?
10. Reusability: Will the application meet one or many user's needs?☐
11. Installation ease: How difficult is the conversion and installation?☐
12. Operational ease: How automated are start-up, backup and recovery procedures?☐
13. Multiple sites: Will the application be installed at multiple sites for multiple organizations?☐
14. Adaptability and flexibility: Is the application specifically designed to facilitate change?☐☐

Function Points: Example of a GSC Rating

GSC	Value(0-5)
Data communications	1
Distributed data processing	1
Performance	4
Heavily used configuration	0
Transaction rate	1
On-Line data entry	0
End-user efficiency	4
On-Line update	0
Complex processing	0
Reusability	3
Installation ease	4
Operational ease	4
Multiple sites	0
Adaptability and Flexibility	0
Total	**22**

Calculation of Effort in COCOMO

- Estimate number of instructions
  - KDSI = "Kilo Delivered Source Instructions"
- Determine project complexity parameters: A, B
  - Regression analysis, matching project data to equation
- 3 levels of difficulty that characterize projects
  - Simple project ("organic mode")
  - Semi-complex project ("semidetached mode")
  - Complex project ("embedded mode")
- Calculate effort
  - Effort = $A * KDSI^B$
- Also called Basic COCOMO

Calculation of Effort in Basic COCOMO

Formula: $Effort = A * KDSI^B$

- Effort is counted in person months: 152 productive hours (8 hours per day, 19 days/month, less weekends, holidays, etc.)
- A, B are constants based on the complexity of the project

Project Complexity	A	B
Simple	2.4	1.05
Semi-Complex	3.0	1.12
Complex	3.6	1.20

Calculation of Development Time

Basic formula: $T = C * Effort^D$

- T = Time to develop in months
- C, D = constants based on the complexity of the project
- Effort = Effort in person months (see slide before)

Project Complexity	C	D
Simple	2.5	0.38
Semi-Complex	2.5	0.35
Complex	2.5	0.32

Basic COCOMO Example

Volume = 30000 LOC = 30KLOC

Project type = Simple

Effort $= 2.4 * (30)^{1.05} = 85$ PM

Development Time $= 2.5 * (85)^{0.38} = 13.5$ months

=> Avg. staffing: 85/13.5 = 6.3 persons

=> Avg. productivity: 30000/85 = 353 LOC/PM

Compare:	Semi-detached:	135 PM	13.9 M	9.7 persons
	Embedded:	213 PM	13.9 M	15.3 persons

Cocomo: Example of Cost Driver Rating

Cost Driver	Very Low	Low	Nominal	High	Very High	Extra High
Required software reliability	0.75	0.88	1.00	1.15	1.40	-
Database size	-	0.94	1.00	1.08	1.16	-
Product Complexity	0.70	0.85	1.00	1.15	1.30	1.65
Execution Time Constraint	-	-	1.00	1.11	1.30	1.66
Main storage constraint	-	-	1.00	1.06	1.21	1.56
Virtual Storage volatility	-	0.87	1.00	1.15	1.30	-
Computer turn around time	-	0.87	1.00	1.07	1.15	-
Analyst capability	1.46	1.19	1.00	0.86	0.71	-
Applications experience	1.29	1.13	1.00	0.91	0.82	-
Programmer Capability	1.42	1.17	1.00	0.86	0.70	-
Virtual machine experience	1.21	1.10	1.00	0.90	-	-
Prog. language experience	1.14	1.07	1.00	0.95	-	-
Use of modern Practices	1.24	1.10	1.00	0.91	0.82	-
Use of software tools	1.24	1.10	1.00	0.91	0.83	-
Required schedule	1.23	1.08	1.00	1.04	1.10	-

Other COCOMO Models

- Intermediate COCOMO
  - 15 cost drivers yielding a multiplicative correction factor
  - Basic COCOMO is based on value of 1.00 for each of the cost drivers
- Detailed COCOMO
  - Multipliers depend on phase: Requirements; System Design; Detailed Design; Code and Unit Test; Integrate & Test; Maintenance

Steps in Intermediate COCOMO

- Basic COCOMO steps:
  - Estimate number of instructions
  - Determine project complexity parameters: A, B
  - Determine level of difficulty that characterizes the project
- New step:
  - Determine cost drivers
    - 15 cost drivers c1 , c1 …. c15
- Calculate effort
  - Effort = A * KDSIB * c1 * c1 …. * c15

Calculation of Effort in Intermediate COCOMO

Basic formula:

Effort = $A * KDSI^B * c_1 * c_1 .... * c_{15}$

Effort is measured in PM (person months, 152 productive hours (8 hours per day, 19 days/month, less weekends, holidays, etc.)

A, B are constants based on the complexity of the project

Project Complexity	A	B
Simple	2.4	1.05
Semi-Complex	3.0	1.12
Complex	3.6	1.20

Intermediate COCOMO: 15 Cost drivers

- Product Attributes
  - Required reliability
  - Database size
  - Product complexity
- Computer Attributes
  - Execution Time constraint
  - Main storage constraint
  - Virtual Storage volatility
  - Turnaround time
- Personnel Attributes
  - Analyst capability
  - Applications experience
  - Programmer capability
  - Virtual machine experience
  - Language experience
- Project Attributes
  - Use of modern programming practices
  - Use of software tools
  - Required development schedule

- Rated on a qualitative scale between "very low" and "extra high"
- Associated values are multiplied with each other.

COCOMO II

- Revision of COCOMO I in 1997
- Provides three models of increasing detail
  - Application Composition Model
    - Estimates for prototypes based on GUI builder tools and existing components
  - Early Design Model
    - Estimates before software architecture is defined
    - For system design phase, closest to original COCOMO, uses function points as size estimation
  - Post Architecture Model
    - Estimates once architecture is defined
    - For actual development phase and maintenance; Uses FPs or SLOC as size measure
- Estimator selects one of the three models based on current state of the project.

COCOMO II (cont'd)

- Targeted for iterative software lifecycle models
  - Boehm's spiral model
  - COCOMO I assumed a waterfall model
    - 30% design; 30% coding; 40% integration and test
- COCOMO II includes new costs drivers to deal with
  - Team experience
  - Developer skills
  - Distributed development
- COCOMO II includes new equations for reuse
  - Enables build vs. buy trade-offs

COCOMO II: Added Cost drivers

- Development flexibility
- Team cohesion
- Developed for reuse
- Precedent
- Architecture & risk resolution
- Personnel continuity
- Documentation match life cycle needs
- Multi-Site development.

How would you reply to this post?

- Post on www.ifpug.org:

  "Our organization has just started to use function point analysis for estimation.

  We have no internal metrics from the past we are not sure what productivity (hours/FP) to use for Cobol projects and for Java projects, in the financial industry.

  Can anyone tell me their experiences with hours/FP for this platform or a place to go where to find this industry metrics?"

L.W.F Factor