# COSC 310: Software Engineering

Dr. Bowen Hui
University of British Columbia Okanagan

## Recall Sizing from PM Lecture

- prediction of **amount of code** needed to fulfill requirements
- estimate size of software
- size measurements in various units
    - lines of code
    - function points / feature points
    - number of processes in a data flow diagram
    - number of objects, attributes, services in an object-oriented diagram
- used as input to other planning steps

## Approaches to Sizing

- we focus on counting:
- **lines of code (LOC)**
  - techniques:
    - intuitive (estimation by analogy)
    - participative (estimation by expert opinion)
    - historical
- **function points (FP)**
  - related to functionality of software
  - technique: FP sizing

# Example: Counting LOC

- 1 public void paint ( Graphics g )
- 2 {
- 3 print (g, “Sequence in original order “, a, 25, 25);
- 4 sort();
- 5 print(g, “results”, a,25, 55);
- 6 }
- 7 public void sort()
- 8 {
- 9 for (int pass=1; pass <a.length; pass++)
- 10 for (int i=0; i<a.length; i++)
- 11 if (a[ i ] > a[ i+1] )
- 12 { hold = a[i];
- 13 a[i] = a[i+1];
- 14 a[i+1]=hold;
- 15 }
- 16 }

```
1public void paint ( Graphics g )
2    {
3 print (g, "Sequence in original
     order ", a, 25, 25); sort();
     print(g, "results", a, 25, 55);
4    }
5    public void sort()
6    {
7    for (int pass=1; pass
     <a.length; pass++)
8    for (int i=0; i<a.length;
     I++)   if (a[ i ] > a[ i+1] )
9    { hold = a[i]; a[i] = a[i+1];
     a[i+1]=hold; }
10   }
```

## Counting LOC Guidelines

- ensure each source code line is counted once
- ignore:
  - comment lines, debug code, temporary code, reused source statements, etc.
  - only delivered source code lines should be counted
- count data definitions, invocations, calls to macros once
- translate LOC to assembly language equivalent lines

## Using LOC to Measure Productivity

- many organizations measure productivity as LOC produced
  - "The average productivity rate remains, over the last 2 decades at about 3000 delivered LOC/programmer per year"

- what's wrong with this?

## Using LOC to Measure Productivity

- many organizations measure productivity as LOC produced
  - "The average productivity rate remains, over the last 2 decades at about 3000 delivered LOC/programmer per year"
- what's wrong with this?
  - # defects
  - efficiency of code / better design
  - relationship between effort and LOC is not linear
    - programmer's skill level
    - conceptual complexity of code
  - does not distinguish between generated vs handwritten code
    - code generators tend to produce excessive code

## Additional Issues

- difficult to accurately estimate early in lifecycle
  - actual LOC not known until near project completion
- only way to predict LOC is by analogy and expert opinion
- higher level languages require fewer LOC than low-level languages

## LOC Strengths

- easy to measure
- widely used and accepted
- availability of automated tools
- directly relates to the end product
  - all software products have them
- lends itself well to re-estimation
- permits comparison between diverse development groups

## LOC Size Calculation

1. Intuitive (by analogy)
   - subjective estimate based on experience
   - compare new software to functionality of existing one and adjust estimate

2. Participative (via expert opinion)

3. Historical (via fuzzy logic)

## LOC Size Calculation

2. Participative (via expert opinion)
   - other project team members are formally involved in estimation process
   - product WBS is used to arrive at duration estimates
   - 3 estimates are derived:
     - optimistic (O) = min value
     - pessimistic (P) = max value
     - realistic (R) = most likely value
   - $LOC = (O + 4R + P) / 6$

## 3. Historical (via fuzzy logic)

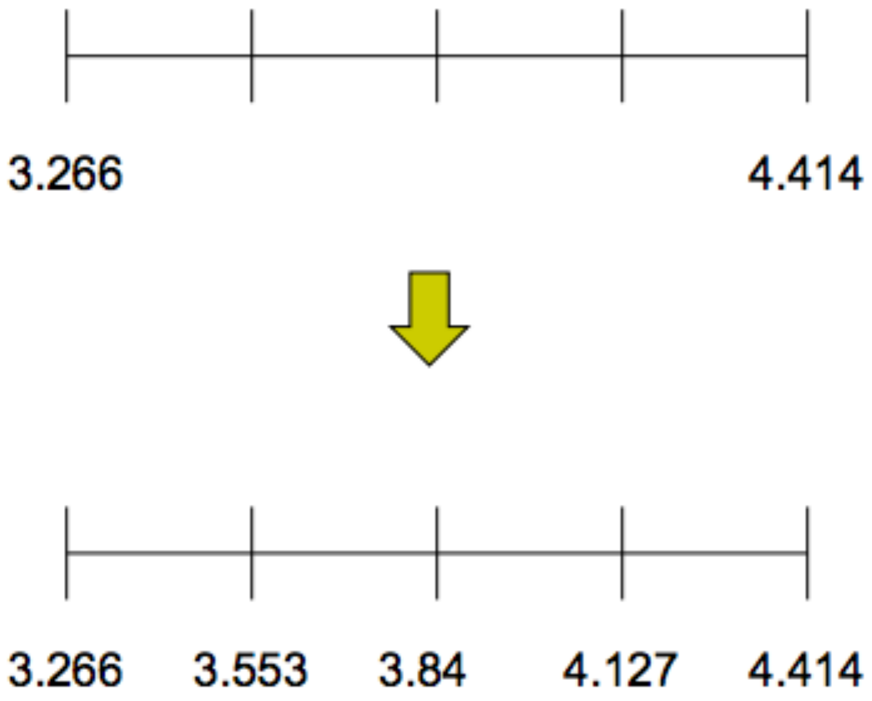
- gather size data from experience
- divide data in 5 size categories
  - very large, large, medium, small, very small
  - establish size ranges
  - include all existing and expected products
- establish min and max ranges
- establish ranges for remaining categories
- allocate available data to the categories
- given new program: judge which category and subcategory it most closely resembles

## Example: Historical Sizing Gathering Data

- You have historical data on 5 programs as follows:
  - a file utility of 1,844 LOC
  - a file management program of 5,834 LOC
  - a personnel record keeping program of 6,845 LOC
  - a report generating package of 18,386 LOC
  - an inventory management program of 25,943 LOC

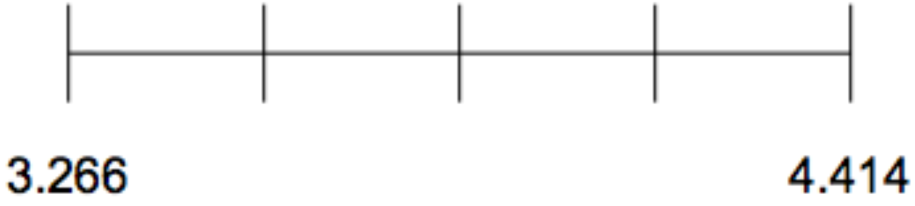
## Example: Historical Sizing (cont.) Establish Ranges

- To establish ranges:
  - $\log_{10}(1844) = 3.266$
  - $\log_{10}(25{,}943) = 4.414$
  - difference is 1.148
  - Start with 4 ranges:
    - 1/4th of 1.148 is 0.287
    - the logs of the ranges are thus spaced 0.287 apart

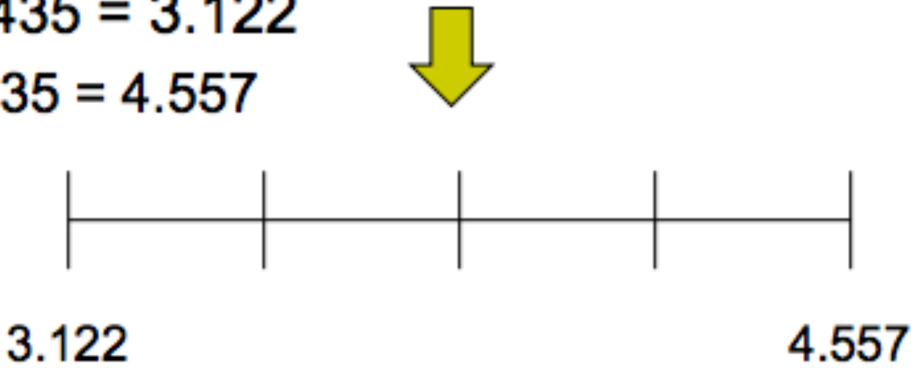


## Example: Historical Sizing (cont.) Extend Lower and Upper Bounds

- To be realistic, increase the lower/upper bounds
  - accommodates for projects at extreme ranges
  - Increase both ends by half of range size (0.287/2=0.1435)

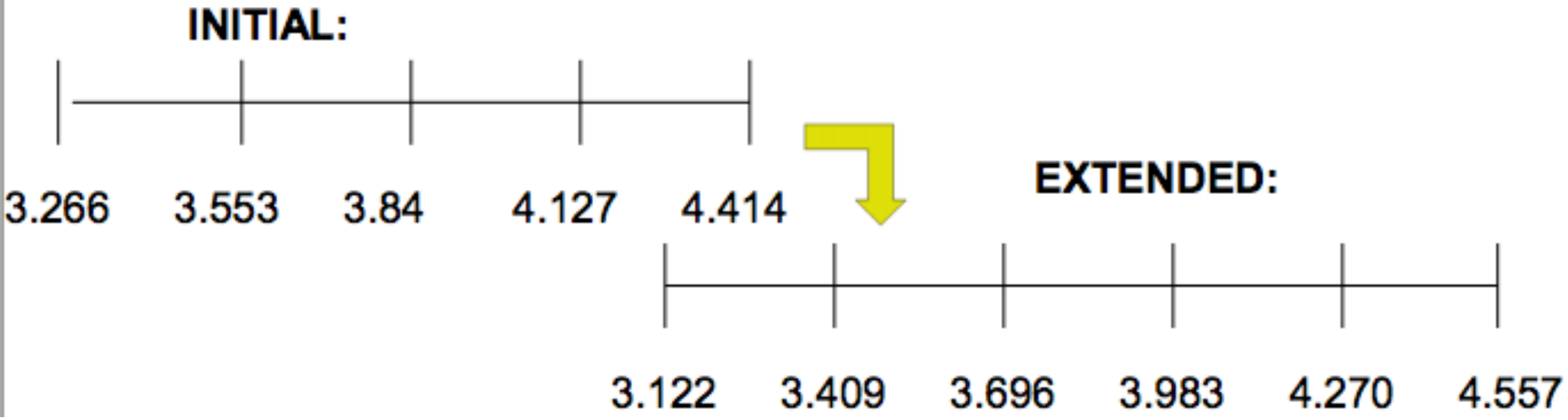


- Lower bound: 3.266 - 0.1435 = 3.122
- Upper bound: 4.414 + 0.1435 = 4.557



# Example: Historical Sizing (cont.) Subdivide Each Range

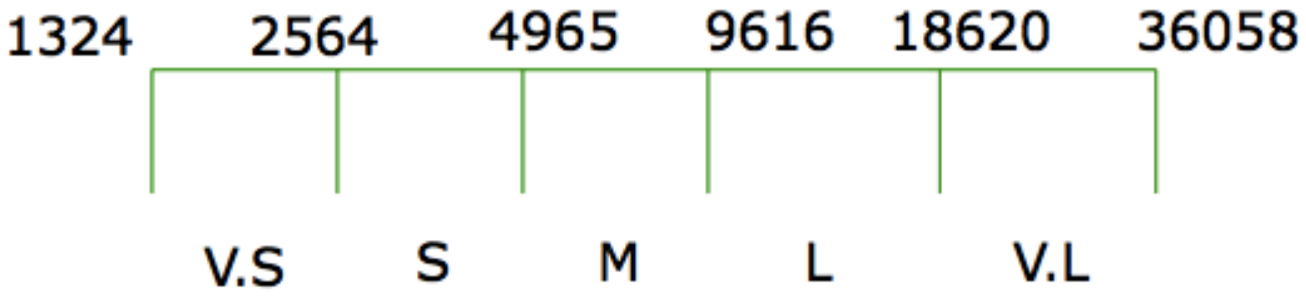
- 3.122 (First Point)
- 3.122 + 0.287 = 3.409 (Second Point)
- 3.409 + 0.287 = 3.696 (Third Point)
- 3.696 + 0.287 = 3.983 (Fourth Point)
- 3.983 + 0.287 = 4.270 (Fifth Point)
- 4.270 + 0.287 = 4.557 (Sixth Point)



# Example: Historical Sizing (cont.) LOC Conversion

- Convert each log value back to LOC
  - Power (10, 3.122) = $10^{3.122}$ = 1324
  - Power (10, 3.409) = $10^{3.409}$ = 2564
  - …
- The ranges are now labeled as Very Small, Small, Medium, Large and Very Large

$y = a^x$
$x = \log_a(y)$



Historical Data – **1844(smallest) and 25943(largest)**

## Example: Historical Sizing (cont.) Allocate Data to Categories

- Previous data:
  - file utility of 1,844 LOC
  - file management program of 5,834 LOC
  - personnel record keeping program of 6,845 LOC
  - report generating package of 18,386 LOC
  - inventory management program of 25,943 LOC
- The 5 size ranges using fuzzy set theory are:

| | |
|---|---|
| V.L - 1,325 to 2,564: | file utility |
| S - 2,564 to 4,965: | none |
| M - 4,965 to 9,616: | file management, |
| | personnel record program |
| L - 9,616 to 18,620: | report generator |
| V.L - 18,620 to 36,058: | inventory management |

**Example: Courtesy Software Engineering Institute at CMU**

## Example: Historical Sizing (cont.) Estimate New Program

- Your new program has the following requirements:
  - analyze marketing performance by product line
  - project the likely sales in each product category
  - allocate these sales to marketing regions and time periods
  - produce a monthly report of these projections and the actual results

## Example: Historical Sizing (cont.) Estimate New Program

- In comparing the new program to the historical data you make the following judgments:
  - it is a substantially more complex application than either the file management or personnel programs
  - it is not as complex as the inventory management program
  - it appears to have significantly more function than the report package

- You conclude that the new program is in the lower end of “very large,” or from 18K - 25K LOC

## Pros and Cons of Historical Sizing

- advantages
  - ?

## Pros and Cons of Historical Sizing

- advantages
  - based on relevant historical data
  - easy to use
  - requires no special tools or training
  - provides reasonably good estimates where new work is like prior experience
- disadvantages
  - ?

## Pros and Cons of Historical Sizing

- disadvantages
  - requires a lot of data
  - estimators must be familiar with previous programs
  - only provides a crude sizing
  - not useful for new program types
  - not useful for programs much larger or much smaller than existing data

## Summary So Far: LOC Approaches

- techniques for counting LOC
  - intuitive
  - expert opinion
  - historical

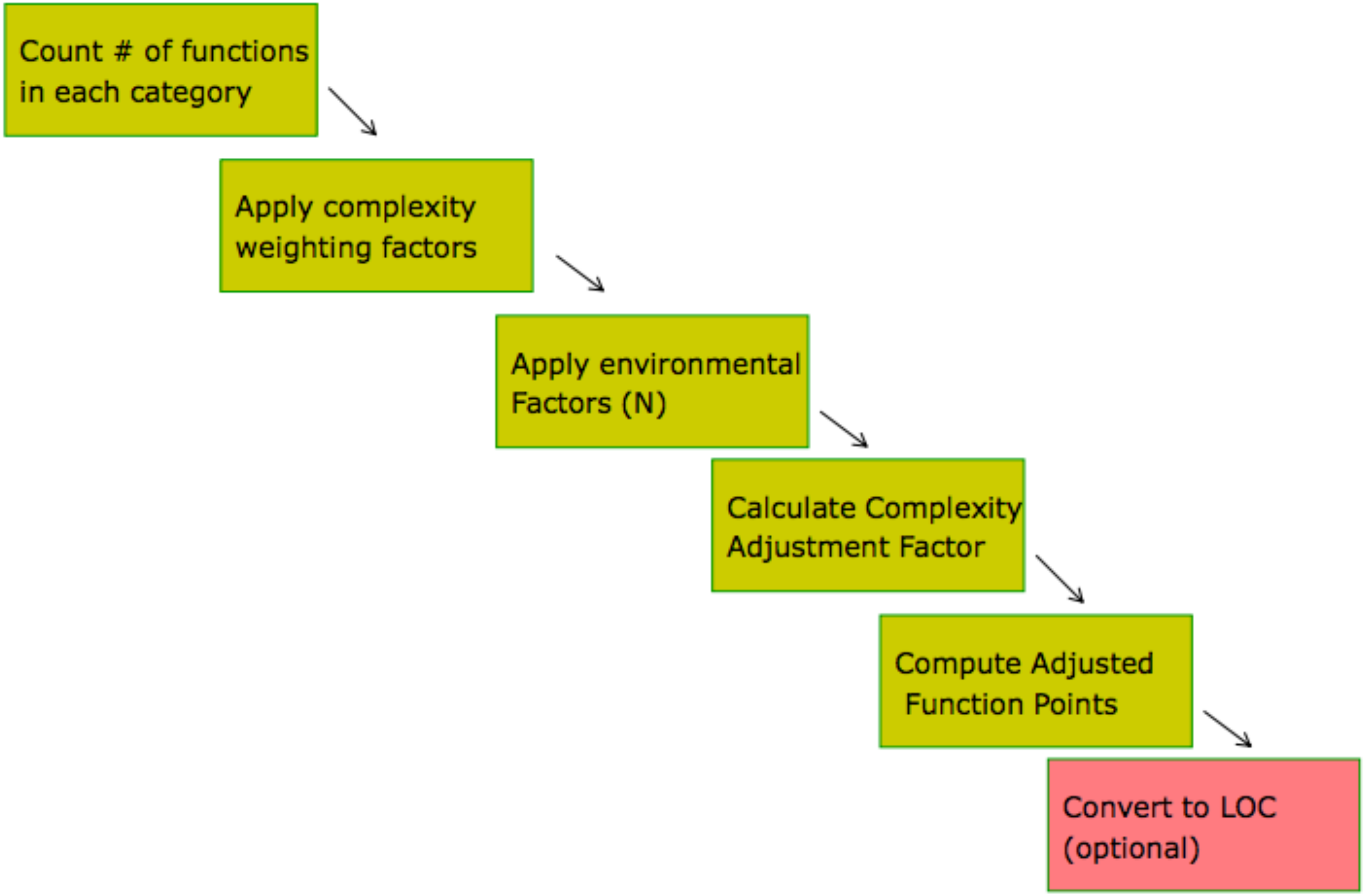
- next: counting function points

## Function Points as a Unit of Size

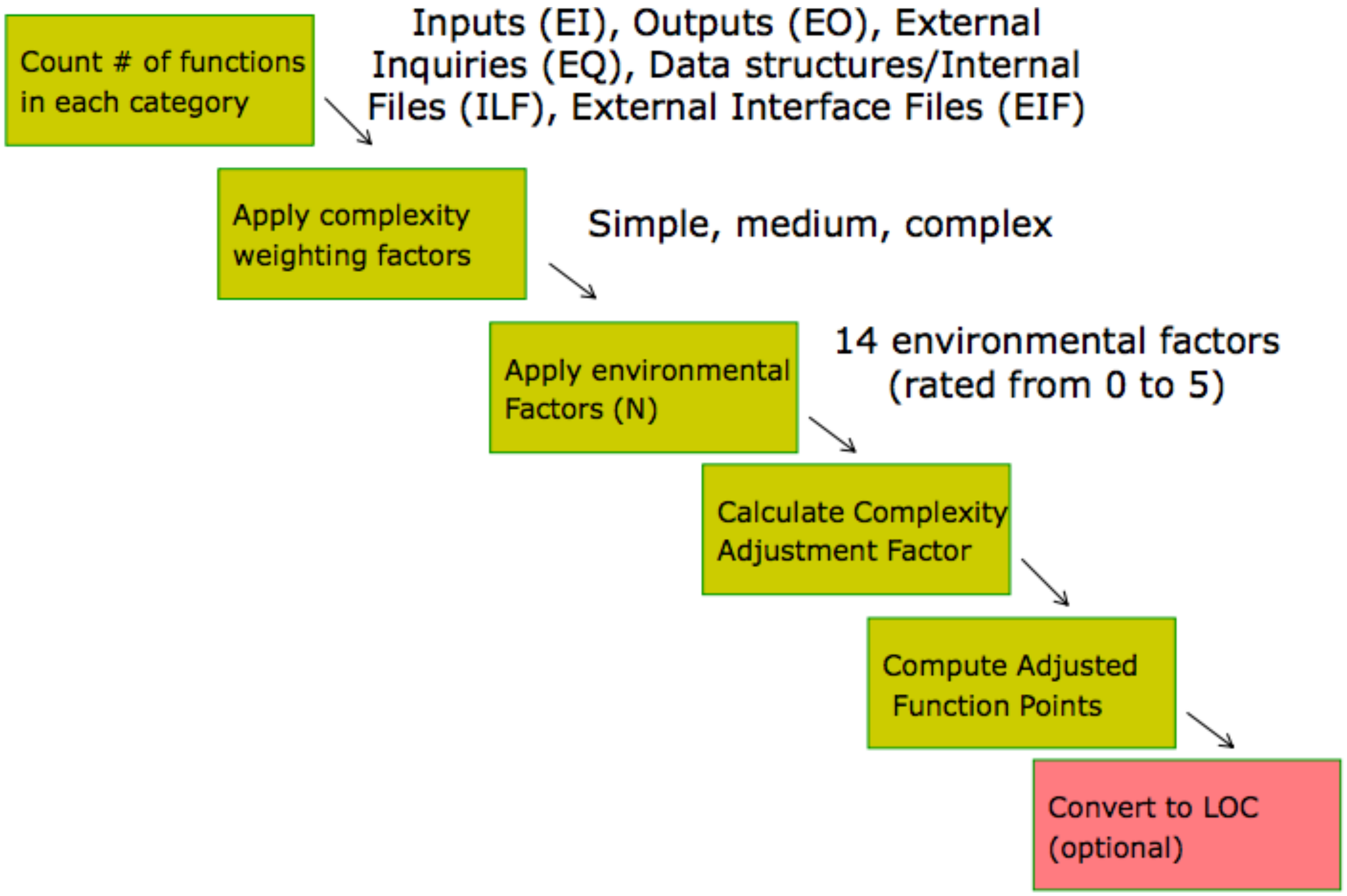
- Function Points (FPs) as unit of measurement
- Accounts for number of functions needed
- Accounts for complexity of functions
- Uses design documents to derive size
  - Data flow diagram
  - Entity-relationship diagram
  - Data dictionary
- Can be used early in the life cycle

- Created by Alan J. Albrecht (IBM) in 1979
- FP counting was standardized by International Function Point user group (IFPUG) in 1986
  - **www.ifpug.org**

## FP Process Overview


Count # of functions in each category
Apply complexity weighting factors
Apply environmental Factors (N)
Calculate Complexity Adjustment Factor
Compute Adjusted Function Points
Convert to LOC (optional)

## FP Process Overview


Count # of functions in each category
Inputs (EI), Outputs (EO), External Inquiries (EQ), Data structures/Internal Files (ILF), External Interface Files (EIF)
Apply complexity weighting factors
Simple, medium, complex
Apply environmental Factors (N)
14 environmental factors (rated from 0 to 5)
Calculate Complexity Adjustment Factor
Compute Adjusted Function Points
Convert to LOC (optional)

# FP Process Overview

Count # of functions in each category

Inputs (EI), Outputs (EO), External Inquiries (EQ), Data structures/Internal Files (ILF), External Interface Files (EIF)

FP

Apply complexity weighting factors

Simple, medium, complex

N

Apply environmental Factors (N)

14 environmental factors (rated from 0 to 5)

CAF = 0.65+ (0.01 x N)

Calculate Complexity Adjustment Factor

AFP = FP x CAF

Compute Adjusted Function Points

LOC = AFP X # of LOC required to implement 1 FP

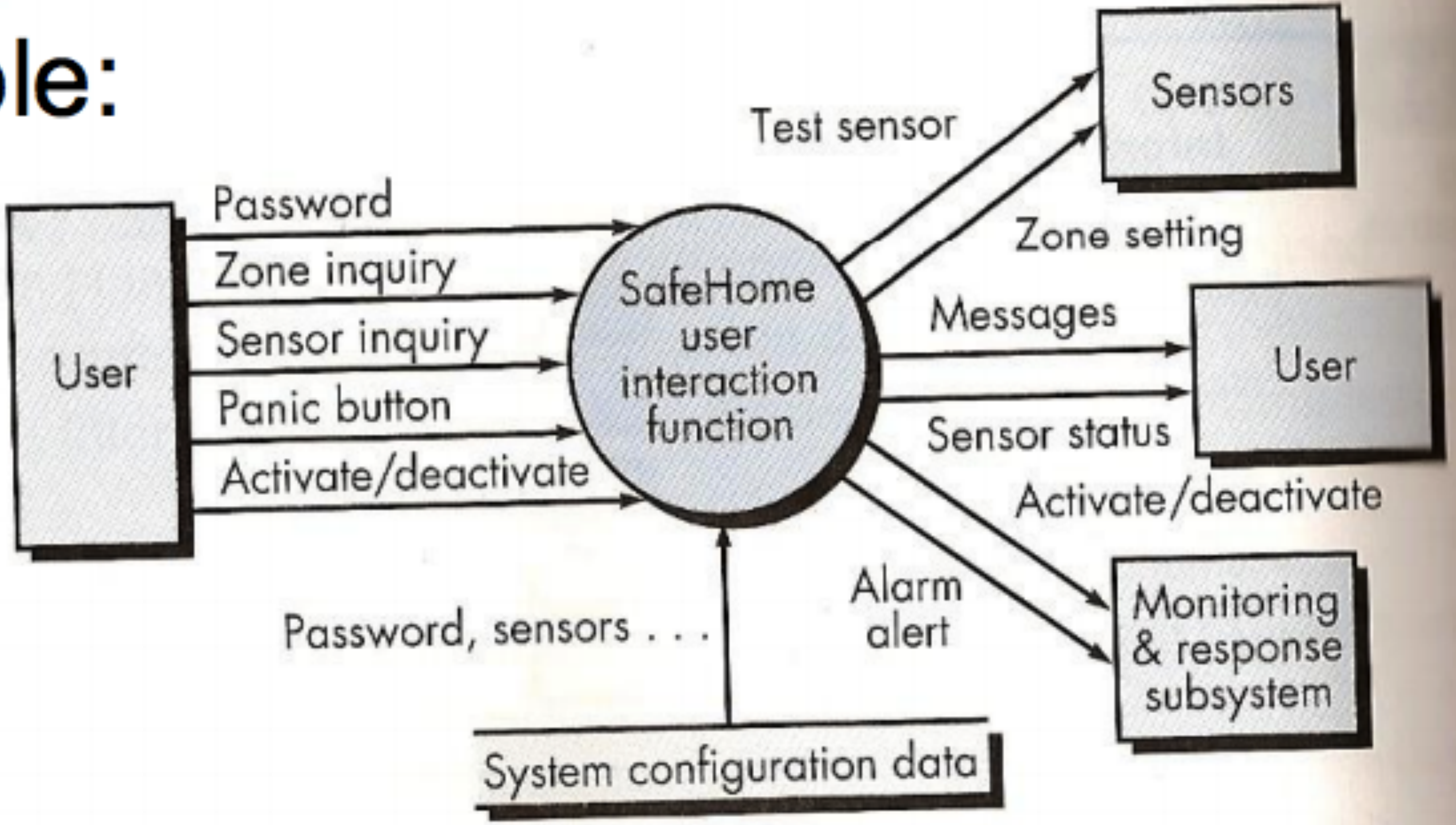
Convert to LOC (optional)

## FP Step 1

- Count the number of software requirement functions
- Best to have **logical design** of system architecture for this

| Type | Examples |
|---|---|
| Inputs | Info units, software input, user input |
| Outputs | Info units, software output, screen data, error messages |
| Inquiries (input and output types) | Direct DB accesses, external requests/commands for software |
| Data structures/ Internal Files | Logical files, primary data structures of user data |
| Interfaces | Data and control shared between multiple system components<br>*** NOT "user interface" |

# Example:



Inputs – password, panic button, activate/deactivate

Outputs – messages and sensor status

Inquiries – zone enquiry/sensor enquiry

DS/Files – System Configuration Data

Interfaces – Zone setting, test sensor (SENSOR SUBSYSTEM), alarm alert, activate/deactivate (MONITORING RESPONSE SYSTEM)

## FP Step 2

- Apply scale weighting factors

| Type | Simple | Average | Complex | Total |
|---|---|---|---|---|
| Inputs | ___X 3 | ___X 4 | ___X 6 | |
| Outputs | ___X 4 | ___X 5 | ___X 7 | |
| Inquiries (input and output types) | ___X 3 | ___X 4 | ___X 6 | |
| Data structures/ Internal Files | ___X 7 | ___X 10 | ___X 15 | |
| Interfaces | ___X 5 | ___X 7 | ___X 10 | |
| | | | Total (FP): Referred to as the "raw" function points | |

## Calculation Example

- You first estimate the numbers of raw function points as follows :

| | |
|---|---|
| 12 inputs with scale of 4: | 12 x 4 = 48 |
| 7 outputs with scale of 5: | 7 x 5 = 35 |
| 0 inquiries with scale of 4: | 0 x 4 = 0 |
| 3 files with scale of 10: | 3 x 10 = 30 |
| 2 interfaces with scale of 7: | 2 x 7 = 14 |

  - total raw function points:
    - FP = (48+35+0+30+14) = 127

## FP Steps 3, 4, 5 Overview

- empirical data shows that environmental factors have a max impact of +/- 35% on FP calculation
- step 3
  - identifies 14 factors total
  - rates each factor based on how they affect development process
  - scale 1-5, with 5=results in more complexity
  - rating of 0 means N/A
- step 4: adjusts raw FP
- step 5: calculates AFP

## FP Steps 3, 4, 5 Overview

- step 4
  - adjusts raw FP (step 2) based on ratings in step 3
  - compute complexity adjustment factor
  - models -35% and +35% variation

- step 5
  - calculates adjusted FP (AFP) = CAF * FP

# Environmental Factors #1-#3

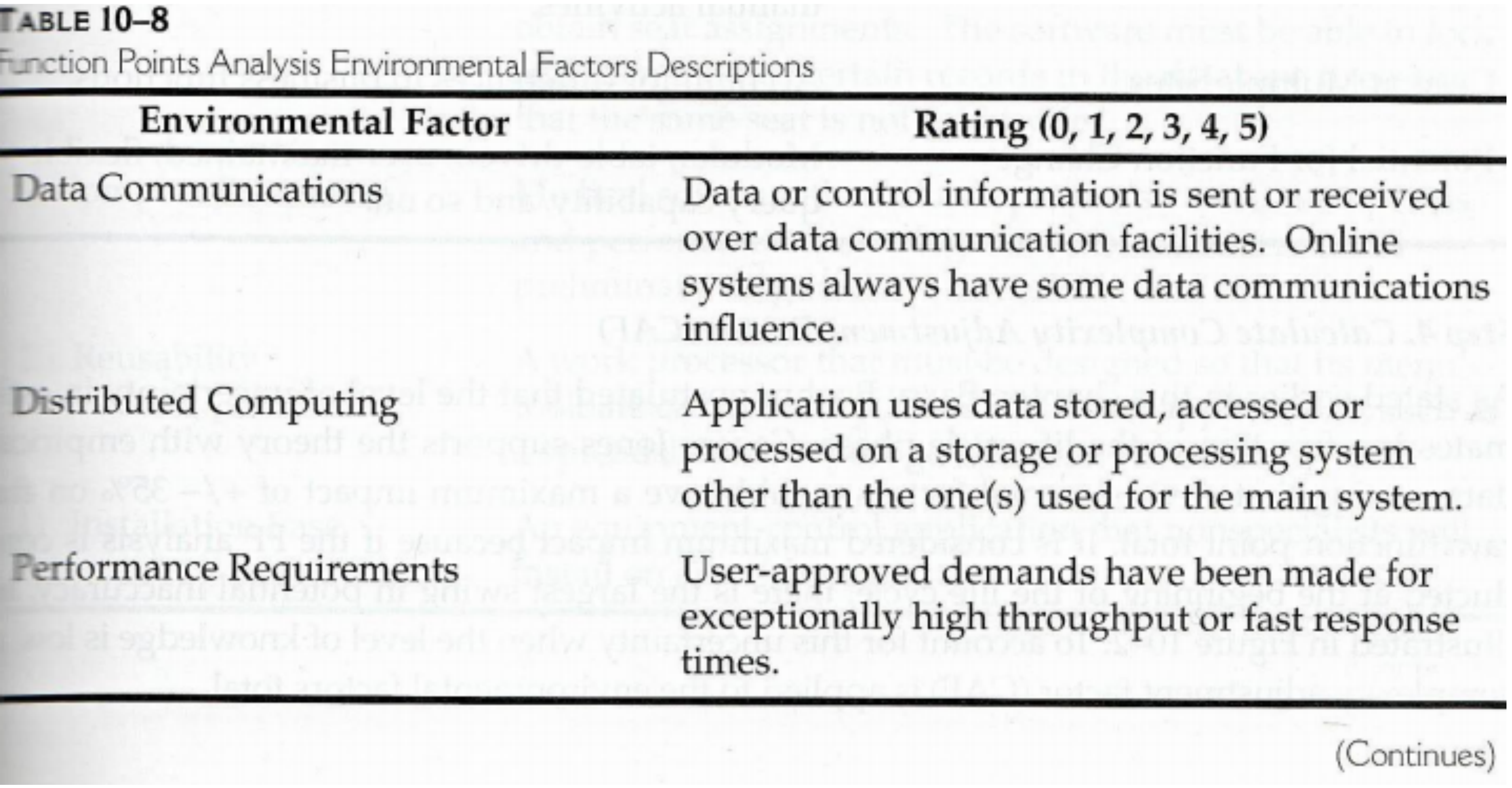


TABLE 10–8

Function Points Analysis Environmental Factors Descriptions

| Environmental Factor | Rating (0, 1, 2, 3, 4, 5) |
|---|---|
| Data Communications | Data or control information is sent or received over data communication facilities. Online systems always have some data communications influence. |
| Distributed Computing | Application uses data stored, accessed or processed on a storage or processing system other than the one(s) used for the main system. |
| Performance Requirements | User-approved demands have been made for exceptionally high throughput or fast response times. |

(Continues)

# Environmental Factors Scoring Examples

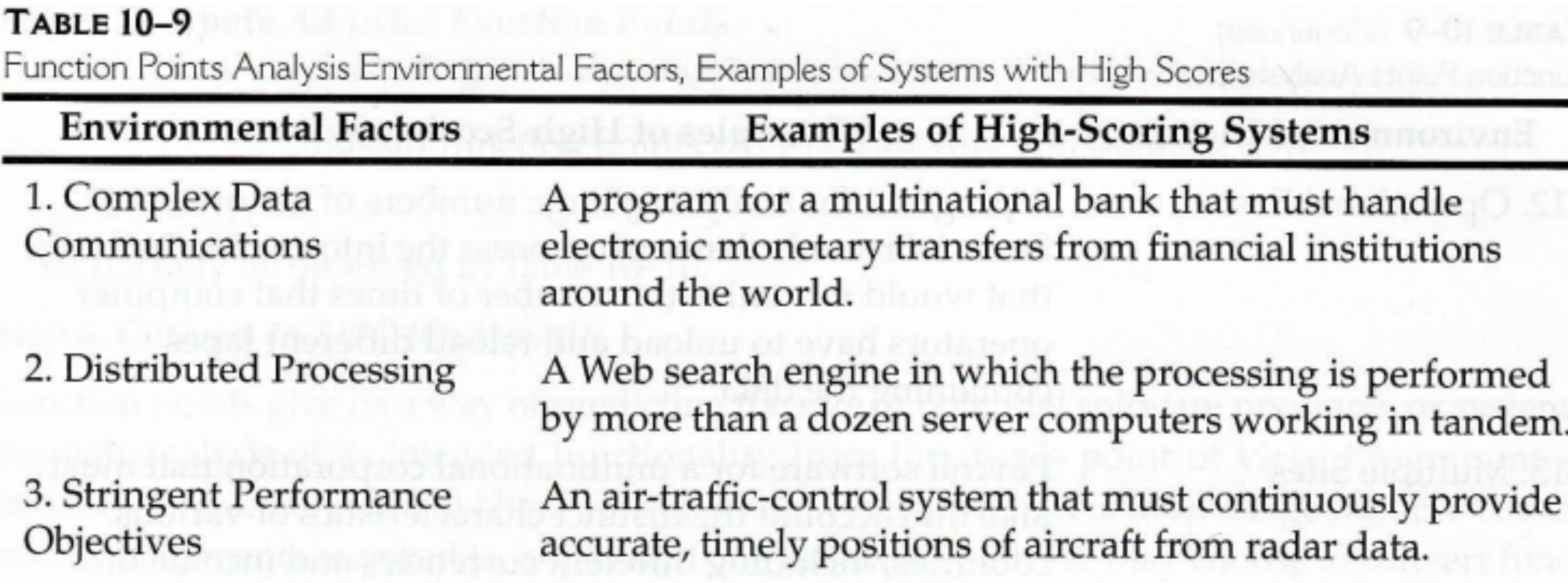


**TABLE 10–9**
Function Points Analysis Environmental Factors, Examples of Systems with High Scores

| Environmental Factors | Examples of High-Scoring Systems |
|---|---|
| 1. Complex Data Communications | A program for a multinational bank that must handle electronic monetary transfers from financial institutions around the world. |
| 2. Distributed Processing | A Web search engine in which the processing is performed by more than a dozen server computers working in tandem. |
| 3. Stringent Performance Objectives | An air-traffic-control system that must continuously provide accurate, timely positions of aircraft from radar data. |

# Environmental Factors #4-#8

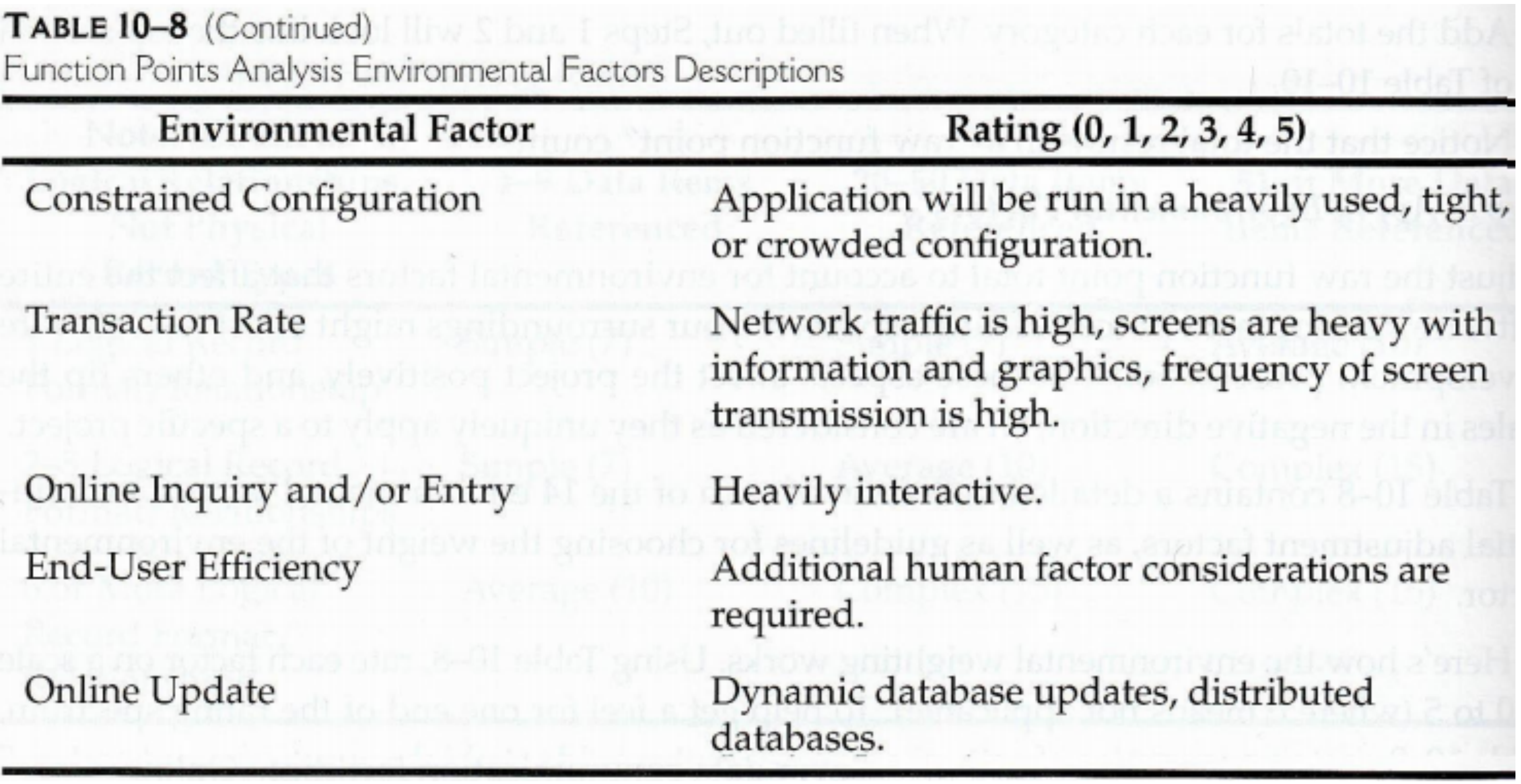


**TABLE 10–8** (Continued)
Function Points Analysis Environmental Factors Descriptions

| Environmental Factor | Rating (0, 1, 2, 3, 4, 5) |
|---|---|
| Constrained Configuration | Application will be run in a heavily used, tight, or crowded configuration. |
| Transaction Rate | Network traffic is high, screens are heavy with information and graphics, frequency of screen transmission is high. |
| Online Inquiry and/or Entry | Heavily interactive. |
| End-User Efficiency | Additional human factor considerations are required. |
| Online Update | Dynamic database updates, distributed databases. |

# Environmental Factors Scoring Examples

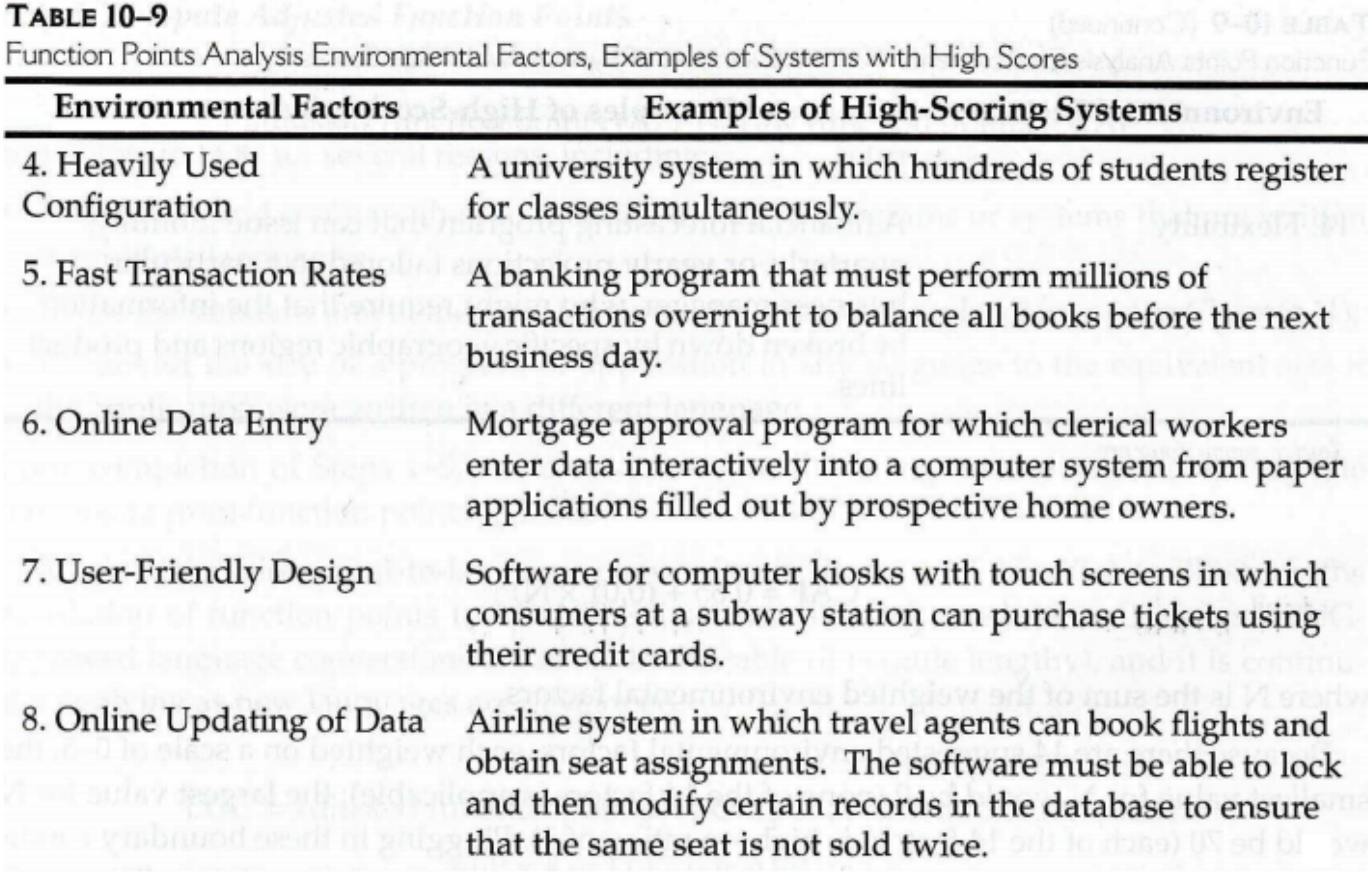


**TABLE 10–9**
Function Points Analysis Environmental Factors, Examples of Systems with High Scores

| Environmental Factors | Examples of High-Scoring Systems |
|---|---|
| 4. Heavily Used Configuration | A university system in which hundreds of students register for classes simultaneously. |
| 5. Fast Transaction Rates | A banking program that must perform millions of transactions overnight to balance all books before the next business day. |
| 6. Online Data Entry | Mortgage approval program for which clerical workers enter data interactively into a computer system from paper applications filled out by prospective home owners. |
| 7. User-Friendly Design | Software for computer kiosks with touch screens in which consumers at a subway station can purchase tickets using their credit cards. |
| 8. Online Updating of Data | Airline system in which travel agents can book flights and obtain seat assignments. The software must be able to lock and then modify certain records in the database to ensure that the same seat is not sold twice. |

## Environmental Factors #9-#14

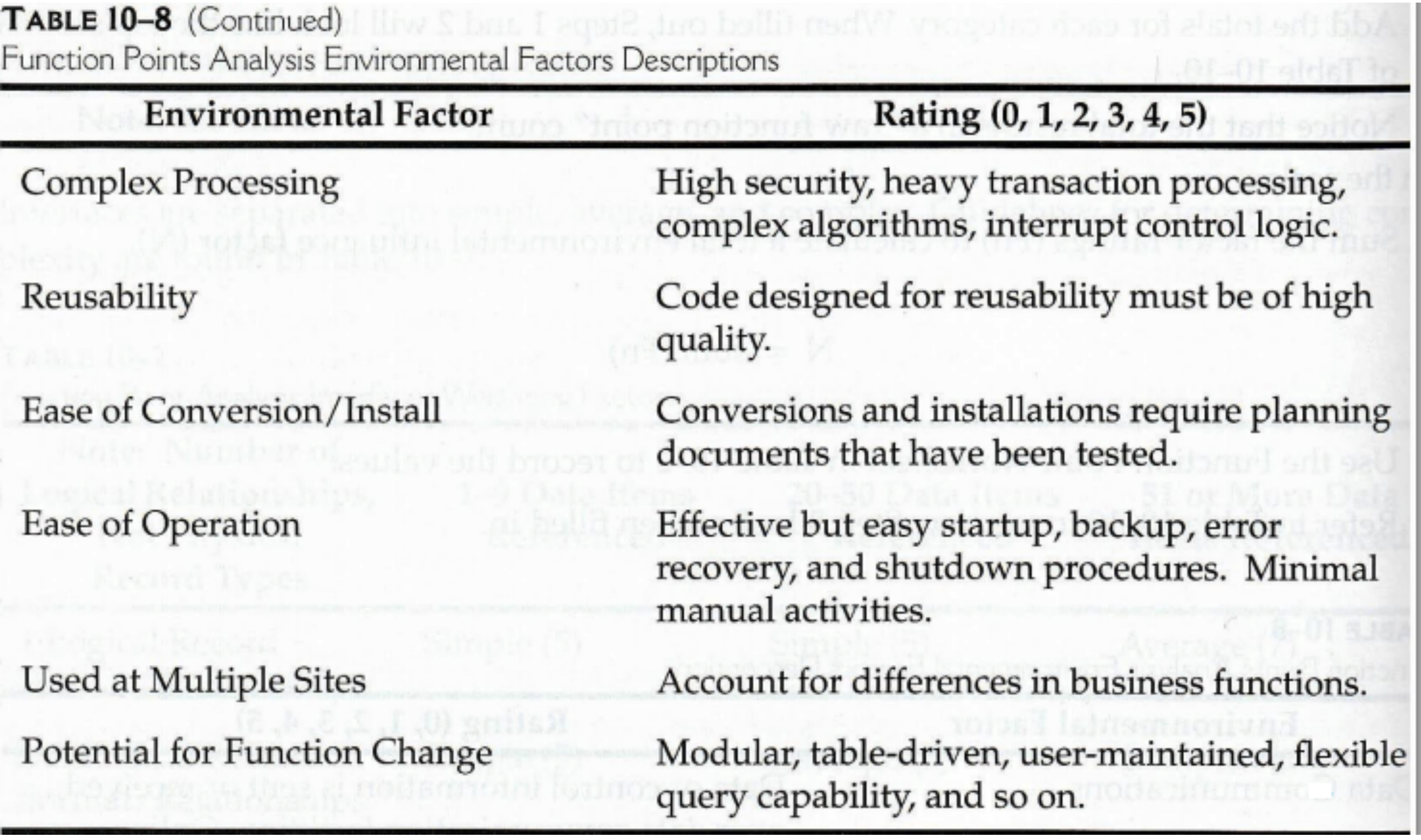


**Table 10–8** (Continued)
Function Points Analysis Environmental Factors Descriptions

| Environmental Factor | Rating (0, 1, 2, 3, 4, 5) |
|---|---|
| Complex Processing | High security, heavy transaction processing, complex algorithms, interrupt control logic. |
| Reusability | Code designed for reusability must be of high quality. |
| Ease of Conversion/Install | Conversions and installations require planning documents that have been tested. |
| Ease of Operation | Effective but easy startup, backup, error recovery, and shutdown procedures. Minimal manual activities. |
| Used at Multiple Sites | Account for differences in business functions. |
| Potential for Function Change | Modular, table-driven, user-maintained, flexible query capability, and so on. |

# Environmental Factors Scoring Examples

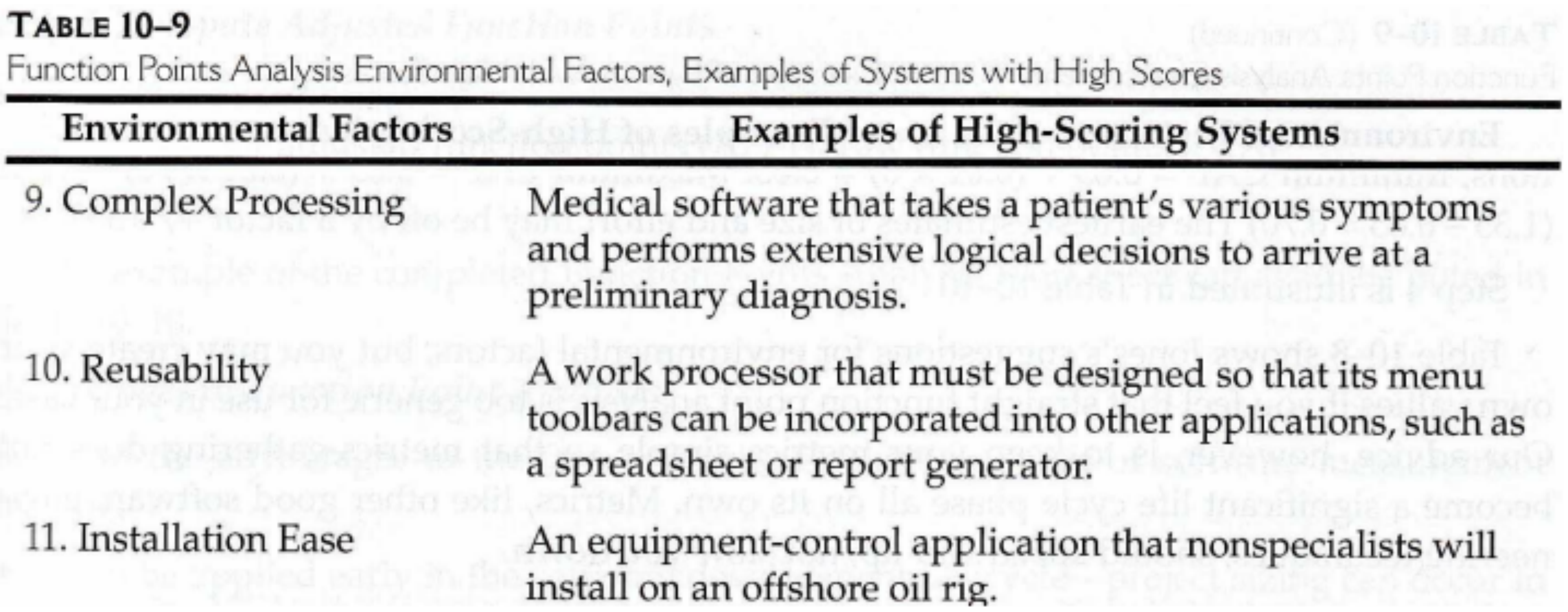


**Table 10–9**
Function Points Analysis Environmental Factors, Examples of Systems with High Scores

| Environmental Factors | Examples of High-Scoring Systems |
|---|---|
| 9. Complex Processing | Medical software that takes a patient's various symptoms and performs extensive logical decisions to arrive at a preliminary diagnosis. |
| 10. Reusability | A work processor that must be designed so that its menu toolbars can be incorporated into other applications, such as a spreadsheet or report generator. |
| 11. Installation Ease | An equipment-control application that nonspecialists will install on an offshore oil rig. |

## Environmental Factors Scoring Examples

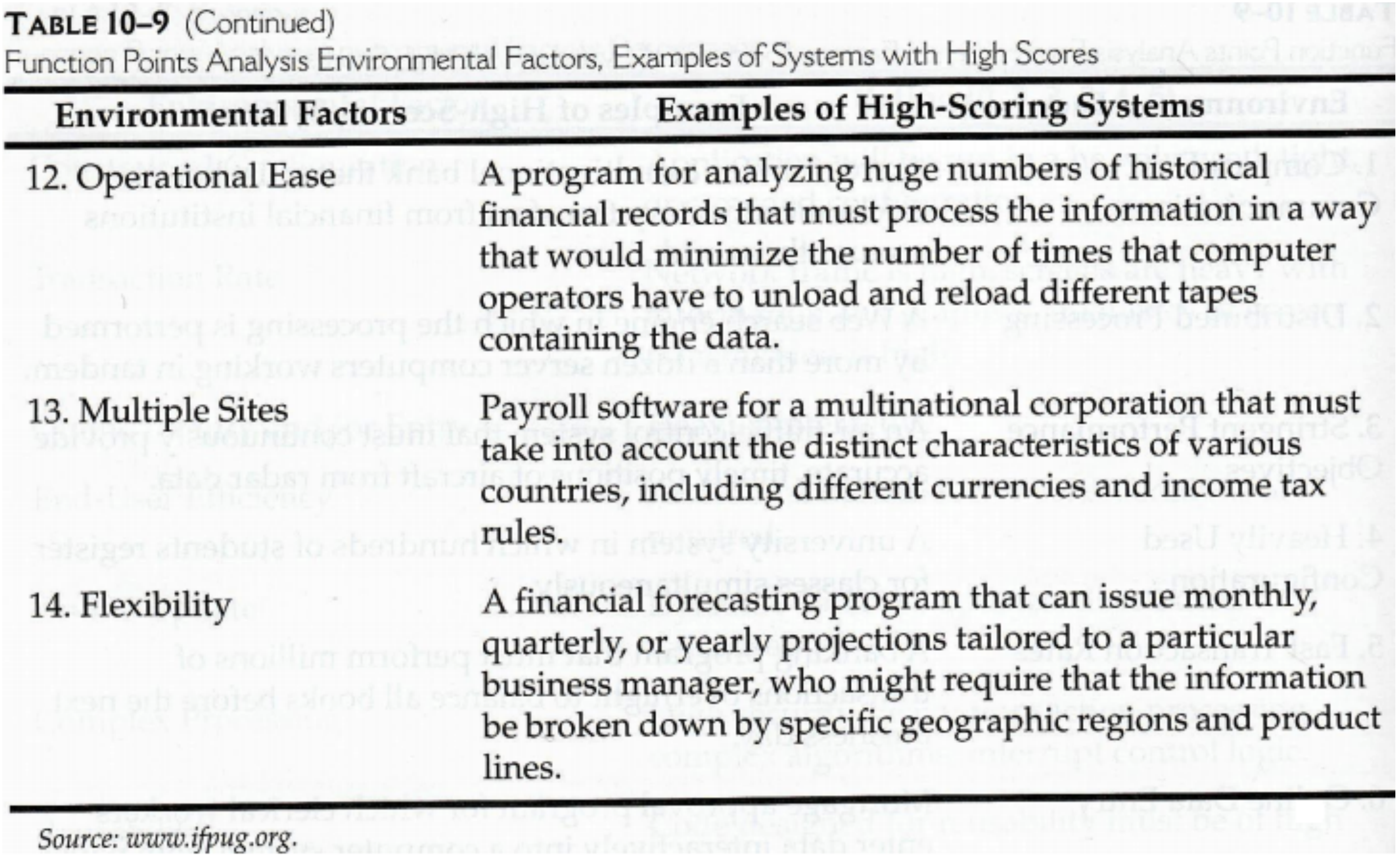


**Table 10–9** (Continued)
Function Points Analysis Environmental Factors, Examples of Systems with High Scores

| Environmental Factors | Examples of High-Scoring Systems |
|---|---|
| 12. Operational Ease | A program for analyzing huge numbers of historical financial records that must process the information in a way that would minimize the number of times that computer operators have to unload and reload different tapes containing the data. |
| 13. Multiple Sites | Payroll software for a multinational corporation that must take into account the distinct characteristics of various countries, including different currencies and income tax rules. |
| 14. Flexibility | A financial forecasting program that can issue monthly, quarterly, or yearly projections tailored to a particular business manager, who might require that the information be broken down by specific geographic regions and product lines. |

*Source: www.ifpug.org.*

## Adjust for Environmental Factors

- step 4
    - calculate complexity adjustment factor
    - CAF = 0.65 + (0.01 * N)
        - where N = sum of the scores of all the environmental factors
        - note: N ranges in [0,70]
        - so CAF ranges in [0.65, 1.35]
- step 5
    - calculate adjusted FPs
    - AFP = CAF * FP

## FP Step 4 and 5

| | | | | | |
|---|---|---|---|---|---|
| C1 | Data Communications | 4 | C8 | On-line Update | |
| C2 | Distrib.Processing | | C9 | Complex Processing | 3 |
| C3 | Performance Critical | | C10 | Reusability | |
| C4 | Heavily Used Config | | C11 | Installation Ease | |
| C5 | Transaction Rate | | C12 | Operation Ease | 5 |
| C6 | On-Line data Entry | 4 | C13 | Multiple Sites | |
| C7 | End-user Efficiency | | C14 | Appl. design. for chg | 5 |
| | | | | | |
| | | | | N: | 21 |

Complexity multiplier
= 0.65 + 0.01 X N
= 0.65 + 0.01 X 21
= 0.86

Adjusted FP
= FP X CAF
=127 X 0.86
=109.22

## FP Step 6 (Optional)

- convert to LOC
  - LOC = AFP * # of LOC per AFP
- why might you want to do this conversion?
- alternative: using historical data on hours per function point, calculate the development time for the project

## Summary:

- techniques for counting LOC
  - intuitive
  - expert opinion
  - historical

- technique for counting function points

- next class: estimation