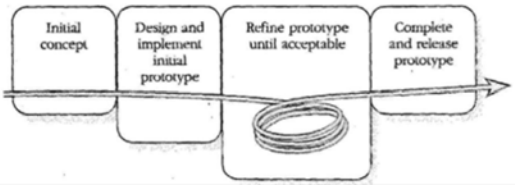
Lecture 4

Software Development (III)

Size estimation & scheduling

1

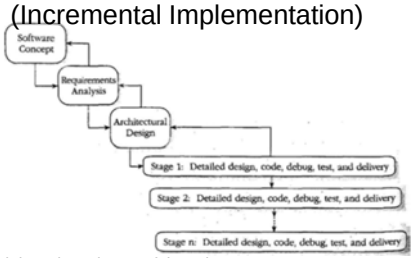
(5) Evolutionary Prototyping



- Begin develop the most visible aspects of the system
- Demonstrate the part to the customer and then continue to develop the prototype based on the feedback
- Repeat the previous step until the prototype is good enough
- Complete the unfinished part of the system
- Very useful when requirements are changing rapidly
- It is impossible to know how long it will take to complete the product

2

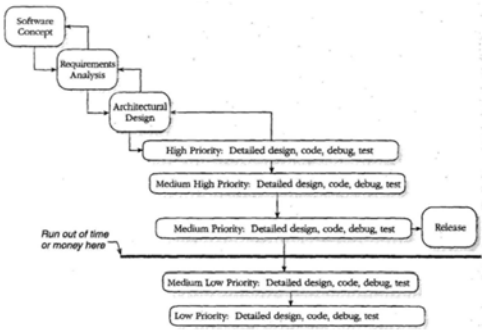
(6) Staged Delivery
(Incremental Implementation)



- Delivers partial products but useful products to the customer
- It will not work without careful planning at both the management and technical levels.
  - Avoid components developed at stage 2 depend on stuff at stage 4.

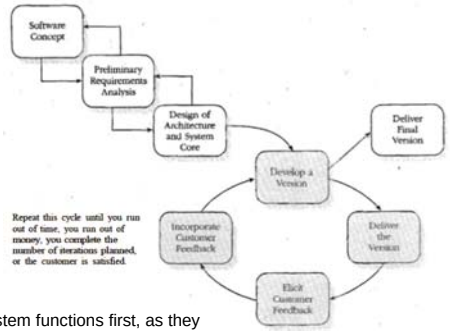
3

(7) Design-to-Schedule



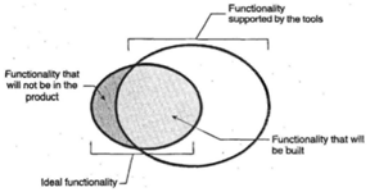
4

(8) Evolutionary Delivery



Deliver core system functions first, as they are unlikely changed by customers

(9) Design-to-Tools



- Tools
  - "code" and "class libraries"
  - Code generators
  - Rapid-development languages
- This model can be combined with other flexible lifecycle models.

6

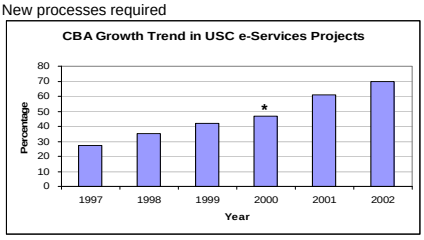
(10) Commercial Off-the-Shelf Software

- Buy commercial software
  - Rarely satisfy the needs
  - Available immediately

7

COTS: The Future Is Here

- Escalate COTS priorities for research, staffing, education
  - Software is not "all about programming" anymore
  - New processes required



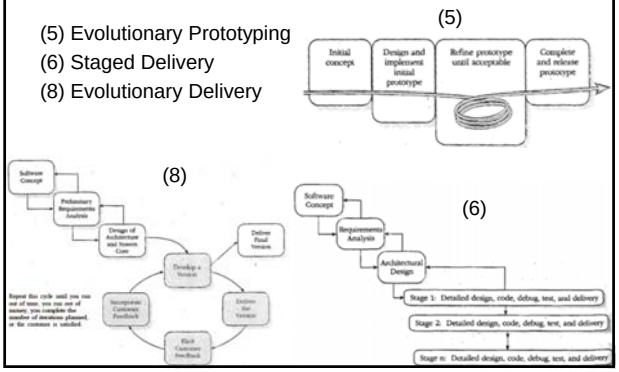
• CBA: COTS-Based Application
* Standish Group CHAOS 2000 (54%)

05/25/06 © USC-CSE 8

Boehm (2006), "A View of 20th and 21st Century Software Engineering", ICSE 2006 Keynote Address

Comparison between (5), (6) and (8)

(5) Evolutionary Prototyping
(6) Staged Delivery
(8) Evolutionary Delivery



Lifecycle model strengths and weaknesses

Lifecycle model capability	Pure Waterfall	Code-and-Fix	Spiral	Modified Waterfalls	Evolutionary Prototyping	Staged Delivery	Evolutionary Delivery	Design-to-Schedule	Design-to-Tools	Commercial Off-the-Shelf Software
Work with poorly understood requirements	Poor	Poor	Excellent	Fair to Excellent	Excellent	Poor	Fair to Excellent	Poor to Fair	Fair	Excellent
Works with poorly understood architecture	Poor	Poor	Excellent	Fair to Excellent	Poor to fair	Poor	Poor	Poor	Poor to Excellent	Poor to Excellent
Produces highly reliable system	Excellent	Poor	Excellent	Excellent	Fair	Excellent	Fair to Excellent	Fair	Poor to Excellent	Poor to Excellent
Produces system with large growth envelope	Excellent	Poor to fair	Excellent	Excellent	Excellent	Excellent	Excellent	Fair to excellent	Poor	N/A
Manage risks	Poor	Poor	Excellent	Fair	Fair	Fair	Fair	Fair to excellent	Poor to fair	N/A
Can be constrained to a predefined schedule	Fair	Poor	Fair	Fair	Poor	Fair	Fair	Excellent	Excellent	Excellent
Has low overhead	Poor	Excellent	Fair	Excellent	Fair	Fair	Fair	Fair	Fair to excellent	Excellent
Allows for midcourse corrections	Poor	Poor to excellent	Fair	Fair	Excellent	Poor	Fair to Excellent	Poor to Fair	Excellent	Poor
Provides customer with progress visibility	Poor	Fair	Excellent	Fair	Excellent	Fair	Excellent	Fair	Excellent	N/A
Provides management with progress visibility	Fair	Poor	Excellent	Fair to Excellent	Fair	Excellent	Excellent	Excellent	Excellent	N/A
Requires little manager or developer sophistication	Fair	Excellent	Poor	Poor to fair	Poor	Fair	Fair	Poor	Fair	Fair

10

Review

- Development Fundamentals
  - Technical Fundamentals
    - Requirement, Design, Software Configuration
  - Quality-Assurance Fundamentals
    - Testing
    - Reviews
- Management Fundamentals
  - Estimation, Scheduling, Planning, Tracking, Measurement
  - *Planning: Life-Cycle Models*

11

Outline

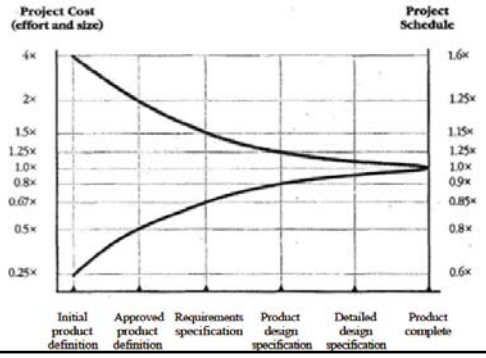
- Estimation
- Accuracy vs. Precision
- Size estimation
- Schedule estimation

12

Estimation

13

Estimate-Convergence Graph



14

Estimate Multipliers by Project Phase

Table 8-1. Estimate Multipliers by Product Phase

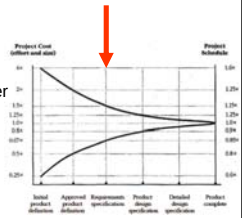
	Effort and Size		Schedule	
Phase	**Optimistic**	**Pessimistic**	**Optimistic**	**Pessimistic**
Initial product concept	**0.25**	**4.00**	**0.60**	**1.60**
Approved product concept	**0.50**	**2.00**	**0.80**	**1.25**
Requirements specification	**0.67**	**1.50**	**0.85**	**1.15**
Product design specification	**0.80**	**1.25**	**0.90**	**1.10**
Detailed design specification	**0.90**	**1.10**	**0.95**	**1.05**

15

RD: Table 8-1

Sample Estimate

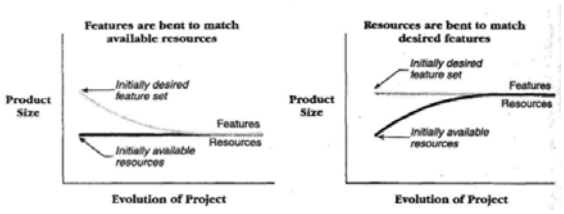
- Project manager estimated 50 man-months
  - Use 10 people → 5 months
  - Use 5 people → 10 months
  - Use 25 people → 2 months
- Completed requirement specification
- 50 x 0.67 = 34 (optimistic)
- 50 x 1.5 = 75 (pessimistic)
- Note the schedule uses different multiplier



16

Estimation vs. Control

- Most customers want more than they can afford



17

Accuracy and Precision (1/2)

- Accuracy
  - how close to the mark
  - a measurement 3 is a more accurate representation of pi than 4 is
- Precision
  - how many digits a measurement has
  - 3.14 is a more precise representation of pi than 3 is
- A measurement can be precise without being accurate, and it can be accurate without being precise.

18

Accuracy and Precision (2/2)

- In software estimation, false precision is the enemy of accuracy.
  - An effort estimate of 40 – 70 man-months might be both the most accurate and the most precise estimate you can make.
  - Simplify it to 55 man-months, that's like representing pi as 3.3232 instead of 3. It looks more precise, but it's really less accurate.
- Shortest-possible software schedules are achieved by creating the most accurate estimates possible, not the most precise. If you want to achieve maximum development speed, avoid false precision.

19

The Process of Estimation

- Estimate the size of the project
  - Number of lines of code
  - Function points
- Estimate the effort (man-months)
- Estimate the schedule (calendar months)
  - Use estimated project size
  - Use historic data

20

Size Estimation

21

Three ways of size estimation

1. Function Points
   - Use an algorithmic approach, such as function points or lines of code, that estimates program size from program features.
2. Program Features
   - Use size-estimation software that estimates program size from your description of program features (screens, dialogs, files, database tables, and so on).
3. Similar Projects
   - If you have already worked on a similar project and know its size, estimate each major piece of the new system as a percentage of the size of a similar piece of the old system. Estimate the total size of the new system by adding up the estimated sizes of each of the pieces.

22

Advantages of Using Function Points

- Technology and platform independence
- Available from early requirements phase
- Consistent and objective unit of measure throughout the life cycle
- Objectively defines software application from the customer perspective
- Objectively defines a series of software applications from the customer's, not the technician's perspective
- Is expressed in terms that users can readily
- understand about their software

23

Size Estimation

The number of function points in a program is based on the number and complexity of each of the following items:

1. **Inputs** — Screens, forms, dialog boxes, controls, or messages through which an end-user or other program adds, deletes, or changes a program's data. This includes any input that has a unique format or unique processing logic.
2. **Outputs** — Screens, reports, graphs, or messages that the program generates for use by an end-user or other program. This includes any output that has a different format or requires a different processing logic than other output types.

24

Size Estimation

3. **Inquiries** — Input/output combinations (input + output) in which an input results in an immediate, simple output.
   - The term originated in the database world and refers to a direct search for specific data, usually using a single key.
   - Queries retrieve data directly from a database and provide only rudimentary formatting, whereas outputs can process, combine, or summarize complex data and can be highly formatted.
4. **Logical internal files** — Major logical groups of end-user data or control information that are completely controlled by the program. A logical file might consist of a single flat file or a single table in a relational database.
5. **External interface files** — Files controlled by other programs with which the program being counted interacts. This includes each major logical group of data or control information that enters or leaves the program.

25

Function-Point Multipliers

Program Characteristic	Function Points		
	Low Complexity	Medium Complexity	High Complexity
Number of inputs	×3	×4	×6
Number of outputs	×4	×5	×7
Inquiries	×3	×4	×6
Logical internal files	×7	×10	×15
External interface files	×5	×7	×10

Source: RD

26

Example

Program Characteristic	Function Points		
	Low Complexity	Medium Complexity	High Complexity
Number of inputs	$6\times3=18$	$2\times4=8$	$3\times6=18$
Number of outputs	$7\times4=28$	$7\times5=35$	$0\times7=0$
Inquiries	$0\times3=0$	$2\times4=8$	$4\times6=24$
Logical internal files	$5\times7=35$	$2\times10=20$	$3\times15=45$
External interface files	$9\times5=45$	$0\times7=0$	$2\times10=20$
Unadjusted function-point total			304
Influence multiplier			1.15
Adjusted function-point total			350

Source: RD

Influence factor ranges between 0.65 – 1.35 to consider factors such as data communication, online data entry, processing complexity, ease of installation, etc.

27

Approximate Language Levels

Language	Level	Statement per Function Point
Excel, Lotus 123, Quattro Pro, etc.	~50	6
Smalltalk 80; Smalltalk/V	15	20
AWK	15	25
Perl	15	25
SAS, SPSS, other statistics packages	10	30
Visual Basic 3	10	30
dBase IV	9	35
Paradox	9	35
Focus	8	40
Oracle	8	40
Sybase	8	40
C++	6.5	50
Quick Basic 3	5.5	60
Lisp	5	65
Ada 83	4.5	70
Modula 2	4	80
Cobol (ANSI 85)	3 - 5	90
Pascal	3.5	90
GW Basic	3.25	100
Fortran 77	3	110
C	2.5	125
Macro assembler	1.5	215
Assembler	1	320

*Adapted from RD

28

Function-Points to LOC Conversion

Approximate Function-Points to Lines-of-Code Conversions

Size in Function Points	Size in Lines of Code					
	Fortran	Cobol	C	C++	Pascal	Visual BASIC
1	110	90	125	50	90	30
100	11,000	9,000	12,500	5,000	9,000	3,000
500	55,000	45,000	62,500	25,000	45,000	15,000
1,000	110,000	90,000	125,000	50,000	90,000	30,000
5,000	550,000	450,000	625,000	250,000	450,000	150,000

Source: RD

29

Size Estimation

- The function point can then be compared to other projects within the same organization or other similar projects.
  - To figure out how many people and how much time is needed.
- The function points can also be used to estimate schedule using Jones's First-Order Estimation Practice (later)
- Size → Effort → Personnel, Schedule, and Cost

30

Project Schedule Estimation

31

Schedule Estimation

- The software is divided into three categories:
  - System: the system software defined here excludes firmware, real-time embedded systems, avionics, process control, etc.
  - Business: business applications such as accounting, reporting, …
  - Shrink-wrap: packaged software, such as Microsoft Office, Adobe creative suite, …
- Three ballpark estimation methods
  1. Schedule-effort equation
  2. Function point → effort
  3. Schedule tables: LOC → effort

32

Schedule Estimation (1/3)

Schedule-Effort Equation

$$Optimal\ Schedule = 3 \times Effort^{1/3}$$

$Optimal\ Schedule: months$
$Effort: man-months$

Assuming an effort estimation gives the effort of 65 man-months
Optimal schedule: 12 months ($3.0 * 65^{1/3}$)
Optimal team size: 5 – 6 team members (65/12)

33

Schedule Estimation (2/3)

Jones's First-Order Estimation

$$Schedule(months) = [Function\ Point]^{power}$$

Kind of Software	Best in Class	Average	Worst in Class
Systems	0.43	0.45	0.48
Business	0.41	0.43	0.46
Shrink-wrap	0.39	0.42	0.45

- Based on the appropriate combination of the kind of software and the organization's class, an exponent can be determined from the above table
- This is an empirical method based on several thousand projects.
- E.g.
  - Developing a business software with an average organization → 0.43
  - $350^{0.43} \sim 12$ calendar months

34

Schedule Estimation (3/3)

Schedule Tables

- Shortest possible schedule. (You can't beat this.)
  - Talent from top 10%, years of experience in environment
  - Ideal management, all staff available day 1
  - Requirements known day 1 and don't change
  - Tools, offices, methods are ideal
- Efficient schedule.
  - Talent from top 25%, low turnover
  - Competent management, staff available as needed
  - Requirements changes are minor (5%); tools, offices are effective
- Nominal schedule
  - Talent from top 50%, turnover 12% per year
  - Some familiarity with tools and environment

35

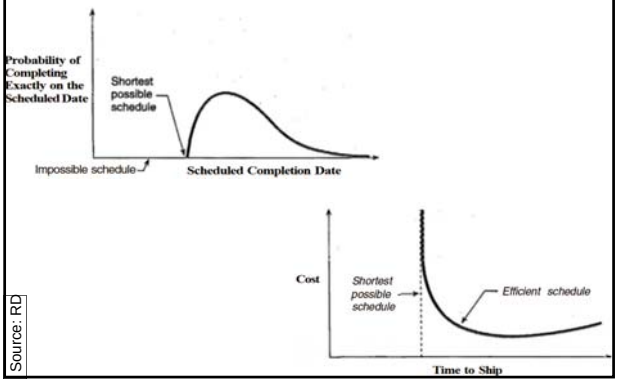
Shortest Possible Schedule Table

	System Products		Business Product		Shrink-Wrap Product	
System Size (lines of code)	Schedule (months)	Effort (man-months)	Schedule (months)	Effort (man-months)	Schedule (months)	Effort (man-months)
10,000	6	25	4	5	4	8
15,000	7	40	4	8	5	13
20,000	8	57	5	11	6	19
25,000	9	74	5	15	6	24
30,000	9	110	6	22	7	37
35,000	10	130	6	26	7	44
40,000	11	170	6	34	7	57
45,000	11	195	6	39	8	66
50,000	11	230	7	46	8	79
60,000	12	285	7	57	9	98
70,000	13	350	8	71	9	120
80,000	14	410	8	83	10	140
90,000	14	480	9	96	10	170
100,000	15	540	9	110	11	190
120,000	16	680	10	140	11	240
140,000	17	820	10	160	12	280
160,000	18	960	10	190	13	335
180,000	19	1,100	11	220	13	390
200,000	20	1,250	11	250	14	440
250,000	22	1,650	13	330	15	580
300,000	24	2,100	14	420	16	725
400,000	27	2,900	15	590	19	1,000
500,000	30	3,900	17	780	20	1,400

Source: RD

36

Shortest possible schedule



Source: RD

Efficient Schedule Table

	System Products		Business Product		Shrink-Wrap Product	
System Size (lines of code)	Schedule (months)	Effort (man-months)	Schedule (months)	Effort (man-months)	Schedule (months)	Effort (man-months)
10,000	8	24	5	5	6	8
15,000	10	38	6	8	7	12
20,000	11	54	7	11	8	18
25,000	12	70	7	14	9	23
30,000	13	97	8	20	9	32
35,000	14	120	8	24	10	39
40,000	15	140	9	30	10	49
45,000	16	170	9	34	11	57
50,000	16	190	10	40	11	67
60,000	18	240	10	49	12	83
70,000	19	290	11	61	13	100
80,000	20	345	12	71	14	120
90,000	21	400	12	82	15	140
100,000	22	450	13	93	15	160
120,000	23	560	14	115	16	195
140,000	25	670	15	140	17	235
160,000	26	709	15	160	18	280
180,000	28	910	16	190	19	320
200,000	29	1,300	17	210	20	360
250,000	32	1,300	19	280	22	470
300,000	34	1,650	20	345	24	590
400,000	38	2,350	22	490	27	830
500,000	42	3,100	25	640	29	1,100

Source: RD

38

Nominal Schedule Table

	System Products		Business Product		Shrink-Wrap Product	
System Size (lines of code)	Schedule (months)	Effort (man-months)	Schedule (months)	Effort (man-months)	Schedule (months)	Effort (man-months)
10,000	10	48	6	9	7	15
15,000	12	76	7	15	8	24
20,000	14	110	8	21	9	34
25,000	15	140	9	27	10	44
30,000	16	185	9	37	11	59
35,000	17	220	10	44	12	71
40,000	18	270	10	54	13	88
45,000	19	310	11	61	13	100
50,000	20	360	11	71	14	115
60,000	21	440	12	88	15	145
70,000	23	540	13	105	16	175
80,000	24	630	14	125	17	210
90,000	25	730	15	140	17	240
100,000	26	820	15	160	18	270
120,000	28	1,000	16	200	20	335
140,000	30	1,200	17	240	21	400
160,000	32	1,400	18	280	22	470
180,000	34	1,600	19	330	23	540
200,000	35	1,900	20	370	24	610
250,000	38	2,400	22	480	26	800
300,000	41	3,000	24	600	29	1,000
400,000	47	4,200	27	840	32	1,400
500,000	51	5,500	29	1,100	35	1,800

Source: RD

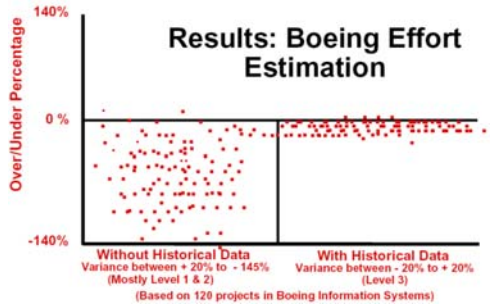
39

Schedule Estimates

- By using function points or lines of code, we can now get a ROUGH/BallPark estimate (better than nothing) on the schedule of the software from the previous two empirical methods.
  - Count FP → Schedule (months) = FP^{power} → # Months = $Effort^{1/3}$ → # people = Effort / Months
  - Count FP → p. 17/18 converts FP to LOC → Use on of the schedule tables on p. 25, 27, or 28 → # of people, # months
- The schedule estimates from these empirical methods exclude the time for requirement specification!
- The best schedule estimate, however, needs to be based on historic performance!

40

Importance of Historical Data



41

Source: CMMI V1.1 and Appraisal Tutorial, Mike Phillips (2004)

Schedule compression equations

$$\text{Schedule Compression Factor} = \frac{\text{Desired Schedule}}{\text{Initial Schedule}}$$

$$\text{Compressed schedule effort} = \frac{\text{Initial effort}}{\text{Schedule compression factor}}$$

Initial schedule estimate: 12
Initial effort estimate: 78 man-months

Want compress schedule to 10 → SCF = 10/12 = 0.83
Compressed schedule effort: 78/SCF = 94 (man-months)

Some researches conclude that it is impossible to have SCF < 0.75 or 0.80
These equations can also be used to "uncompress" a schedule

42

Review: Estimation and Scheduling

- LOC: Lines of Code
- FP: Function Points
- Counting FP
  - $FP^{(0.43)}$ → calendar months → Schedule-Effort Equation → effort → team size
  - FP → LOC → Schedule Tables → effort, calendar months
- Applying estimation-convergence graph yields a ranged-estimation

43

Summary

Project Estimation: size, effort, schedule
size: LOC, FP
effort: man-months
schedule: months

Schedule Pressure & Feature Set Control

44