Identifier	Total Loc	Methods	Type
Component H01	76	1	I/O
Component H02	116	4	Calculation
Component H03	113	7	I/O
Component H04	103	5	Calculation
Component H05	105	4	I/O
Component H06	48	7	Calculation
Component H07	102	2	I/O
Component H08	111	4	I/O
Component H09	128	3	Calculation
Component H10	93	3	I/O
Component H11	133	2	I/O
Component H12	95	8	Calculation
Component H13	67	4	Data
Component H14	113	4	Data
Component H15	102	6	I/O
Component H16	106	4	I/O
Component H17	83	4	Calculation
Component H18	25	1	Data
Component H19	140	7	Calculation
Component H20	72	3	Data
Component H21	10	0	Data

This represents the software components that have been previously written.

Identifier	Total Loc	Methods	LOC/method	ln(LOC/method)
Component H01	76	1	76.0	4.33
Component H02	116	4	29.0	3.37
Component H03	113	7	16.1	2.78
Component H04	103	5	20.6	3.03
Component H05	105	4	26.3	3.27
Component H06	48	7	6.9	1.93
Component H07	102	2	51.0	3.93
Component H08	111	4	27.8	3.32
Component H09	128	3	42.7	3.75
Component H10	93	3	31.0	3.43
Component H11	133	2	66.5	4.20
Component H12	95	8	11.9	2.47
Component H13	67	4	16.8	2.82
Component H14	113	4	28.3	3.34
Component H15	102	6	17.0	2.83
Component H16	106	4	26.5	3.28
Component H17	83	4	20.8	3.03
Component H18	25	1	25.0	3.22
Component H19	140	7	20.0	3.00
Component H20	72	3	24.0	3.18
~~Component H21~~	~~10~~	~~0~~		

average=	3.23
std=	0.56

This column -- LOC/method -- normalizes each component to a standard measurement.

Calculate the natural log of the number of LOC per method.

Calculate the average and standard deviation of the natural log of the values

It is possible to have a component with zero methods (e.g., a class that contains only constants). In such a case, the zero-method component should be ignored.

Identifier	**Total Loc**	**Methods**	LOC/method	ln(LOC/method)
Component H01	76	1	76.0	4.33
Component H02	116	4	29.0	3.37
Component H03	113	7	16.1	2.78
Component H04	103	5	20.6	3.03
Component H05	105	4	26.3	3.27
Component H06	48	7	6.9	1.93
Component H07	102	2	51.0	3.93
Component H08	111	4	27.8	3.32
Component H09	128	3	42.7	3.75
Component H10	93	3	31.0	3.43
Component H11	133	2	66.5	4.20
Component H12	95	8	11.9	2.47
Component H13	67	4	16.8	2.82
Component H14	113	4	28.3	3.34
Component H15	102	6	17.0	2.83
Component H16	106	4	26.5	3.28
Component H17	83	4	20.8	3.03
Component H18	25	1	25.0	3.22
Component H19	140	7	20.0	3.00
Component H20	72	3	24.0	3.18
~~Component H21~~	~~10~~	~~0~~		

average= 3.23
std= 0.56

The size matrix characterizes historical components. The "lower" and "upper" columns are used to designate the relative size of existing components; the "mid" column is used to size new components.

	Low	Mid	Upper
VS	0	8	11
S	11	14	19
M	19	25	33
L	33	44	58
VL	58	77	big

The "lower" column is as follows:
VS=0
S=e^(average-1.5*std)
M=e^(average-0.5*std)
L=e^(average+0.5*std)
VL=e^(average+1.5*std)
All values are **rounded** to the nearest integer.

The "mid" column is calculated along a ln-normal distribution:
VS=e^(average-2*std)
S=e^(average-std)
M=e^(average)
L=e^(average+std)
VL=e^(average+2*std)
All values are **rounded** to the nearest integer.

The "upper" column is as follows:
VS=e^(average-1.5*std)
S=e^(average-0.5*std)
M=e^(average+0.5*std)
L=e^(average+1.5*std)
VL= big
All values are **rounded** to the nearest integer.

Identifier	**Total Loc**	**Methods**	**Type**	**LOC/method**	**ln(LOC/method)**	Relative Size
Component H01	76	1	I/O	76.0	4.33	VL
Component H02	116	4	Calculation	29.0	3.37	M
Component H03	113	7	I/O	16.1	2.78	S
Component H04	103	5	Calculation	20.6	3.03	M
Component H05	105	4	I/O	26.3	3.27	M
Component H06	48	7	Calculation	6.9	1.93	VS
Component H07	102	2	I/O	51.0	3.93	L
Component H08	111	4	I/O	27.8	3.32	M
Component H09	128	3	Calculation	42.7	3.75	L
Component H10	93	3	I/O	31.0	3.43	M
Component H11	133	2	I/O	66.5	4.20	VL
Component H12	95	8	Calculation	11.9	2.47	S
Component H13	67	4	Data	16.8	2.82	S
Component H14	113	4	Data	28.3	3.34	M
Component H15	102	6	I/O	17.0	2.83	S
Component H16	106	4	I/O	26.5	3.28	M
Component H17	83	4	Calculation	20.8	3.03	M
Component H18	25	1	Data	25.0	3.22	M
Component H19	140	7	Data	20.0	3.00	M
Component H20	72	3	Data	24.0	3.18	M
~~Component H21~~	~~10~~	~~0~~	~~Data~~			

average= 3.23
std= 0.56

	Low	Mid	Upper
VS	1	8	11
S	11	14	19
M	19	25	33
L	33	44	58
VL	58	77	big

The "upper" and "lower" columns of the size matrix are used to tag each historical component with a relative size. For example, "M" components are those that have greater than or equal to 19 lines of code per method and less than 33 lines of code per method. Since ComponentH01 has 76 lines of code per method, it is considered VL.

Note that the "upper" lip of L and the "lower" lip of VL are the same. We are taking a conservative approach to estimation and thus always going with the larger size in cases where the historical part falls exactly on the lip. For example, a 33 LOC/method component would be considered L, not M.

Architecture

Component Name:	Component1
Design Approach:	Functional
Parent Component:	
Component Type:	Calculation
Collaborators:	Component2, Component3, Component4
Operations:	Component1

Component Name:	Component2
Design Approach:	Object-oriented
Parent Component:	
Component Type:	Calculation
Collaborators:	
Operations:	op1, op2

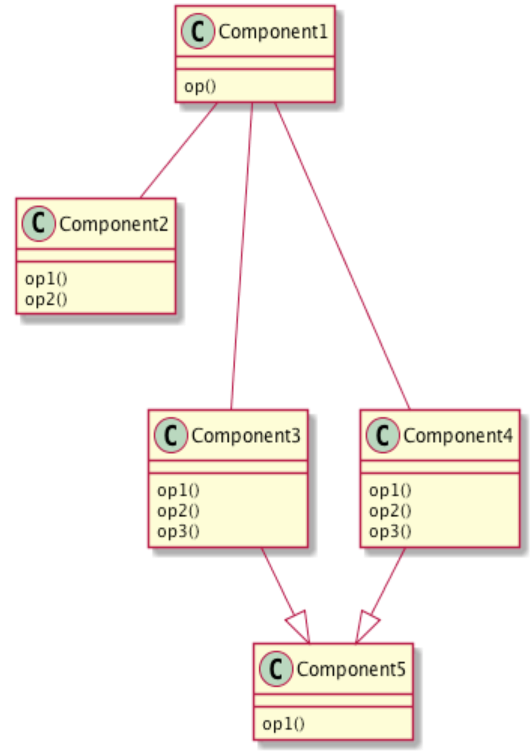
Component Name:	Component3
Design Approach:	Object-oriented
Parent Component:	Component5
Component Type:	I/O
Collaborators:	
Operations:	op1 op2 op3

Component Name:	Component4
Design Approach:	Object-oriented
Parent Component:	Component5
Component Type:	I/O
Collaborators:	
Operations:	op1 op2 op3

Component Name:	Component5
Design Approach:	Object-oriented
Parent Component:	
Component Type:	I/O
Collaborators:	
Operations:	op1

This represents an architectural design consisting of five components.

If you were to sketch the components as class diagrams, they would like like this. Note that we **aren't** using UML; this is provided just to illustrate the relationship among the CRC cards.



Architecture

Component Name:	Component1
Design Approach:	Functional
Parent Component:	
Component Type:	Calculation
Collaborators:	Component2, Component3, Component4
Operations:	Component1

Component Name:	Component2
Design Approach:	Object-oriented
Parent Component:	
Attributes (optional):	
Component Type:	Calculation
Collaborators:	
Operations:	op1, op2

Component Name:	Component3
Design Approach:	Object-oriented
Parent Component:	Component5
Attributes (optional):	
Component Type:	I/O
Collaborators:	
Operations:	op1 op2 op3

Component Name:	Component4
Design Approach:	Object-oriented
Parent Component:	Component5
Attributes (optional):	
Component Type:	I/O
Collaborators:	
Operations:	op1 op2 op3

Component Name:	Component5
Design Approach:	Object-oriented
Parent Component:	
Attributes (optional):	
Component Type:	I/O
Collaborators:	
Operations:	op1

			Changes to existing code				New Code			
Architecture	Estimation Basis	Relative Size	Modified LOC	Added LOC	Deleted LOC	Base LOC	New Methods	LOC/ method	New LOC	
Component1	Compontent H19	M	5	10	14	15	0	0	0	15
Component2	Component H12	S	0	15	25	15	1	14	14	29
Component3										
Component4										
Component5										
									LOCr =	44

Let's suppose that Component1 and Component2 have been identified as having been previously built (and are thus considered coming from "base" code).

Suppose further that we have identified ComponentH19 as being the base code for Component1. After looking over ComponentH19, we expect the we will remove 14 LOC, modify 5 LOC, and add 10 LOC. Although we record the number of lines deleted, this is for completeness only. We will count the number of added and modified lines as contrtibuting to effort.

Along a similar vein, we have determined that Component2 can be built from ComponentH12 by removing 25 LOC and adding 15 LOC. Suppose further that an entirely new method has to be written. Since ComponentH12 is small, we can estimate the size of the new method as being small, meaning, 14 lines of code. The total added amount of code is 15 + 14 = 29

Architecture

Component Name:	Component1
Design Approach:	Functional
Parent Component:	
Component Type:	Calculation
Collaborators:	Component2, Component3, Component4
Operations:	Component1

Component Name:	Component2
Design Approach:	Object-oriented
Parent Component:	
Attributes (optional):	
Component Type:	Calculation
Collaborators:	
Operations:	op1, op2

Component Name:	Component3
Design Approach:	Object-oriented
Parent Component:	Component5
Attributes (optional):	
Component Type:	I/O
Collaborators:	
Operations:	op1 op2 op3

Component Name:	Component4
Design Approach:	Object-oriented
Parent Component:	Component5
Attributes (optional):	
Component Type:	I/O
Collaborators:	
Operations:	op1 op2 op3

Component Name:	Component5
Design Approach:	Object-oriented
Parent Component:	
Attributes (optional):	
Component Type:	I/O
Collaborators:	
Operations:	op1

			Changes to existing code				New Code			
Architecture	Estimation Basis	Relative Size	Modified LOC	Added LOC	Deleted LOC	Base LOC	New Methods	LOC/ method	New LOC	
Component1	Compontent H19	M	5	10	14	15	0	0	0	15
Component2	Component H12	S	0	15	25	15	1	14	14	29
Component3	Component H03	S	0	0	0	0	3	14	42	42
Component4	Component H15	S	0	0	0	0	3	14	42	42
Component5	Component H08	M	0	0	0	0	1	25	25	25
									LOCr =	153

Suppose that the first totally new part has been identified as being most similar to ComponentH03. Since ComponentH03 is a small component, the new component is likely to be small as well. According to the size matrix, small components have 14 lines of code per method, therefore the new component is estimated to have 3*14=42 lines of code.

This concept is repeated for the other components.

We can see from this that we estimate that we are going to put 153 line of code worth of effort into the project.

Identifier	LOCr	LOCp	LOCa	Ea
Project1	163	163	200	600
Project2	301	301	240	840
Project3	134	134	150	420
Project4	293	240	350	2100
Project5	63	106	107	360
Project6	122	154	178	420
Project7	63	114	40	120
Project8	183	199	201	960
Project9	210	224	175	600
Project10	249	251	280	780
Project11	161	177	283	780
Project12	210	230	227	600
Project13	105	136	89	300

LOCr= 153

Size Calculation

sum(LOCa)/sum(LOCr)= 1.12

LOCp= 171

LOCr is the estimated number of lines of code. It represents all the new code that is to be written. It is the sum of the new parts, code added to base components, and code modified from the base.

We hypothesize that the relationship between our historical estimated lines of code and our historical actual lines of code is our best indicator of how to adjust LOCr to be more realistic. This is calcuated as sum(LOCa)/sum(LOCr). This represents an approximation of the slope of the line that models LOCr x LOCa.

LOCp, the planned size, is calculated as LOCr*sum(LOCa)/sum(LOCr), ceiling'ed up to the next integer.

We see here that we plan to write 171 lines of code.

Identifier	LOCr	LOCp	LOCa	Ea	LOCa/LOCr
Project1	163	163	200	600	1.23
Project2	301	301	240	840	0.80
Project3	134	134	150	420	1.12
Project4	293	240	350	2100	1.19
Project5	63	106	107	360	1.70
Project6	122	154	178	420	1.46
Project7	63	114	40	120	0.63
Project8	183	199	201	960	1.10
Project9	210	224	175	600	0.83
Project10	249	251	280	780	1.12
Project11	161	177	283	780	1.76
Project12	210	230	227	600	1.08
Project13	105	136	89	300	0.85
				min=	0.63
LOCr=		153		max=	1.76

Size Calculation

sum(LOCa)/sum(LOCr)=	1.12
LOCp=	171
LPI=	97
UPI=	269
confidence=	0.70 Medium

To determine the confidence we can place in our planned size, we need to find the biggest overestimate and the biggest underestimate.

The lower prediction interval (LPI) is calculated as LOCr*min; the upper prediction interval (UPI) is calculated as LOCr*max. LPI is **floor'ed** and UPI is **ceiling'ed**. If LPI<0, then LPI is set to 1. If UPI<LOCp, then UPI is set to LOCp.

Confidence is quantifed by calculating the square of the correlation of LOCr and LOCa. It is assigned a qualitative value as follows:
r^2 > .75 = HIGH
0.5 < r^2 ≤ .75 = MEDIUM
r^2 ≤ 0.5 = LOW

Identifier	LOCr	LOCp	LOCa	Ea	LOCa/LOCr	Ea/LOCa
Project1	163	163	200	600	1.23	3.00
Project2	301	301	240	840	0.80	3.50
Project3	134	134	150	420	1.12	2.80
Project4	293	240	350	2100	1.19	6.00
Project5	63	106	107	360	1.70	3.36
Project6	122	154	178	420	1.46	2.36
Project7	63	114	40	120	0.63	3.00
Project8	183	199	201	960	1.10	4.78
Project9	210	224	175	600	0.83	3.43
Project10	249	251	280	780	1.12	2.79
Project11	161	177	283	780	1.76	2.76
Project12	210	230	227	600	1.08	2.64
Project13	105	136	89	300	0.85	3.37
				min=	0.63	2.36
LOCr=		153		max=	1.76	6.00

Size Calculation

sum(LOCa)/sum(LOCr)=	1.12
LOCp=	171
LPI=	97
UPI=	269
confidence=	0.70 Medium

Effort Calculation

Prod=	17 LOC/hr
sum(Ea)/sum(LOCa)=	3.52
Ep=	603
LPI=	403
UPI	1026
confidence=	0.69 Medium

To determine the confidence we can place in our planned effort, we need to find the worst overestimate and the worst underestimate.

We hypothesize that the relationship between historical actual size and time best describes the mapping of planned size into planned effort. We obtain planned effort by multiplying sum(Ea)/sum(LOCa) by planned size. Put in other words,
Ep=sum(Ea)/sum(LOCa)*LOCp

LPI, UPI, and confidence are calculated in a similar fashion to size using Ea and LOCa. Because Ep is derived from LOCp, LPI and UPI for effort should also use LOCp. LPI=floor(min(Ea/LOCa)*LOCp) and UPI=ceiling(max(Ea/LOCa)*LOCp). LPI should be set to Ep if LPI is originally calculated as being more then Ep, or is calculated as being negative. SImilarly, UPI should be set to Ep if it is originally calculated as being less than Ep.

Productivity is calculated as the average number of actual LOC per hour over all projects. This value, by convention, is expressed in hours, rounded to the nearest integer.

We start by calculating the square of the correlation of Ea and LOCa. It is assigned a qualitative value as follows:
r^2 > .75 = HIGH
0.5 < r^2 ≤ .75 = MEDIUM
r^2 ≤ 0.5 = LOW

The confidence level we give is the minimum of the confidences of effort and size. For example, if size confidence is low and effort confidence is high, then effort confidence is recorded as low (since we use size to calculate effort).

Identifier	LOCr	LOCp	LOCa	Ea	LOCa/LOCr	Ea/LOCa
Project1	163	163	200	600	1.23	3.00
Project2	301	301	240	840	0.80	3.50
Project3	134	134	150	420	1.12	2.80
Project4	293	240	350	2100	1.19	6.00
Project5	63	106	107	360	1.70	3.36
Project6	122	154	178	420	1.46	2.36
Project7	63	114	40	120	0.63	3.00
Project8	183	199	201	960	1.10	4.78
Project9	210	224	175	600	0.83	3.43
Project10	249	251	280	780	1.12	2.79
Project11	161	177	283	780	1.76	2.76
Project12	210	230	227	600	1.08	2.64
Project13	105	136	89	300	0.85	3.37
			min=		0.63	2.36
LOCr=		153	max=		1.76	6.00

<u>Size Calculation</u>

sum(LOCa)/sum(LOCr)=	1.12	
LOCp=	171	
LPI=	97	
UPI=	269	
confidence=	0.70	Medium

<u>Effort Calculation</u>

Prod=	17	LOC/hr
sum(Ea)/sum(LOCa)=	3.52	
Ep=	603	
LPI=	403	
UPI	1026	
confidence=	0.69	Medium
Best case LPI =	229	
Worst case UPI =	1614	

The LPI and UPI represent our primary estimation range.

Our secondary estimation range is bounded in the best case by the biggest overestimation and the fastest development, and in the worst case by the biggest underestimation and the worst development. The actual calculations are

best case LPI = LOCr * min(LOCa/LOCr) * min(Ea/LOCa)

worst case UPI = LOCr * max(LOCa/LOCr) * max(Ea/LOCa)

COCOMO: The following values represent COCOMO coefficients of your team.

	Very Low	Low	Nominal	High	Very High	Extra High
Scale Factors						
Precedent	6.20	4.96	3.72	2.48	1.24	0.00
Flexibility	5.07	4.05	3.04	2.03	1.01	0.00
Risk resolution	7.07	5.65	4.24	2.83	1.41	0.00
Team Cohesion	5.48	4.38	3.29	2.19	1.10	0.00
Process Level	7.80	6.24	4.68	3.12	1.56	0.00
Effort Multipliers						
Product attributes						
Required software reliability	0.75	0.88	1.00	1.15	1.40	1.40
Size of application database	0.94	0.94	1.00	1.08	1.16	1.16
Complexity of the product	0.70	0.85	1.00	1.15	1.30	1.65
Hardware attributes						
Run-time performance constraints	1.00	1.00	1.00	1.11	1.30	1.66
Memory constraints	1.00	1.00	1.00	1.06	1.21	1.56
Volatility of the virtual machine environment	0.87	0.87	1.00	1.15	1.30	1.30
Required turnabout time	0.87	0.87	1.00	1.07	1.15	1.15
Personnel attributes						
Analyst capability	1.46	1.19	1.00	0.86	0.71	0.71
Applications experience	1.29	1.13	1.00	0.91	0.82	0.82
Software engineer capability	1.42	1.17	1.00	0.86	0.70	0.70
Virtual machine experience	1.21	1.10	1.00	0.90	0.90	0.90
Programming language experience	1.14	1.07	1.00	0.95	0.95	0.95
Project attributes						
Application of software engineering methods	1.24	1.10	1.00	0.91	0.82	0.82
Use of software tools	1.24	1.10	1.00	0.91	0.83	0.83
Required development schedule	1.23	1.08	1.00	1.04	1.10	1.10

$PM = A \times Size^{(B + 0.01 \times \sum SF)} \prod EM$

where,
- PM = planned effort
- Size = LOCp / 1000
- SF = scale factors
- A = calibration data = 2.94
- B = calibration data = 0.91
- EM = Effort multipliers