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

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

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

We want to characterize the original list into 5 values representing a very small value, a small value, etc.
vs=e^(average-2*std)
s=e^(average-std)
m=e^(average)
l=e^(average+std)
vs=e^(average+2*std)
All values are ceiling'ed to the nearest integer.

Note that the values associated with vs, s, m, l, and vl don't necesarily appear in the original list. They just represent a statistical characterization of the original list.

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

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	9	11
S	11	15	20
M	20	26	34
L	34	45	59
VL	59	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 **ceiling'ed** 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 **ceiling'ed** 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 **ceiling'ed** 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

average=	3.23
std=	0.56

	Low	Mid	Upper
VS	0	9	11
S	11	15	20
M	20	26	34
L	34	45	59
VL	59	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 20 lines of code per method and less than 34 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 34 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

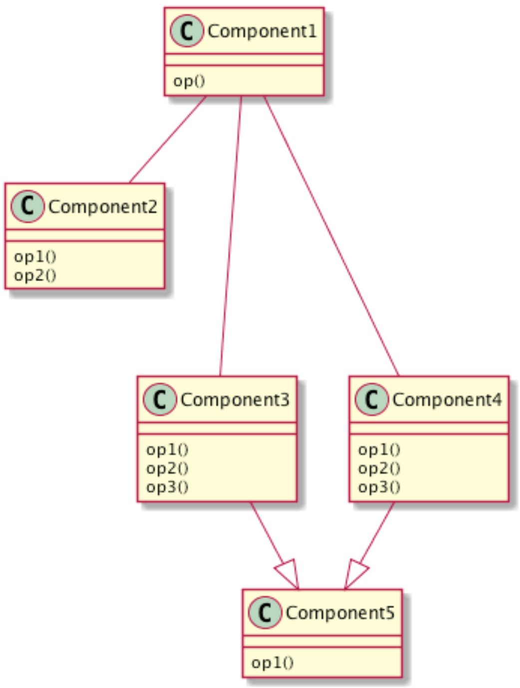
Component Name:	Component3
Design Approach:	Object-oriented
Parent Component:	Component5
Component Type:	I/O
Collaborators:	
Operations:	op1 op2 op3

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.



Base Parts

Base Part	Base (B)	Deleted (D)	Modified (M)	Added (BA)
Component1	140	14	5	10
Component2	95	25	0	30

New Components

New Development	Methods	Similar To	Relative Size	LOC
Component3				
Component4				
Component5				

Let's suppose that two components have been identified as having been previously built (and are coming from "base" code).

Suppose further that we have identified ComponentH19 as being the base code for Component1. This means Component1 will start with a base LOC count of 140. After looking over ComponentH19, we expect the we will remove 14 LOC, modify 5 LOC, and add 10 LOC.

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, 15 lines of code. The total added amount of code is 15 + 15 = 30

Base Parts

Base Part	Base (B)	Deleted (D)	Modified (M)	Added (BA)
Component1	140	14	5	10
Component2	95	25	0	30
subtotals:			5	40

New Components

New Development	Methods	Similar To	Relative Size	LOC
Component3	3	ComponentH03	S	45
Component4	3	ComponentH15	S	45
Component5	1	ComponentH08	M	26
subtotal:				116
			LOCr=	161

Suppose that the first 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, large components have 15 lines of code per method, therefore the new component is estimated to have 3*15=45 lines of code.

This concept is repeated for the other components.

We can see from this that we are going to have to put the following effort into this project (we are assuming that deleting lines of code requires no discernable effort):

New additions to base code: 10 + 30= 40
Modified code*: 5 + 0 = 5
New parts: 45 + 45 + 26 = 116

Thus, LOCr = 40 + 5 + 116= 161 LOC

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= 161

Size Calculation

sum(LOCa)/sum(LOCr)= 1.12

LOCp= 180

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 180 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=		161		max=	1.76

Size Calculation

sum(LOCa)/sum(LOCr)=	1.12
LOCp=	180
LPI=	102
UPI=	283
confidence=	0.70 Medium

To determine the confidence we can place in our planned size, we need to find the worst overestimate and the worst 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=		161	max=		1.76	6.00

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

Size Calculation

sum(LOCa)/sum(LOCr)=	1.12
LOCp=	180
LPI=	102
UPI=	283
confidence=	0.70 Medium

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

Effort Calculation

Prod=	17 LOC/hr
sum(Ea)/sum(LOCa)=	3.52
Ep=	635
LPI=	424
UPI	1080
confidence=	0.69 Medium

LPI, UPI, and confidence are calculated in a similar fashio using Ea and LOCa. Because Ep is derived from LOCp, L for effort should also use LOCp. LPI=floor(min(Ea/LOCa) UPI=ceiling(max(Ea/LOCa)*LOCp). LPI should be set to l originally calculated as being more then Ep, or is calculate negative. SImilarly, UPI should be set to Ep if it is original 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 a
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 c is low and effort confidence is high, then effort confidence is recorded as low (since we use size to calcu

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=		161		max=	1.76	6.00

Size Calculation

sum(LOCa)/sum(LOCr)=	1.12	
LOCp=	180	
LPI=	102	
UPI=	283	
confidence=	0.70	Medium

Effort Calculation

Prod=	17	LOC/hr
sum(Ea)/sum(LOCa)=	3.52	
Ep=	635	
LPI=	424	
UPI	1080	
confidence=	0.69	Medium
Best case LPI =	241	
Worst case UPI =	1698	

The LPI and UPI represent our primary estimation range.

Our secondary estimation range is bounded in the best case b
the biggest overestimation and the fastest development, and i
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)

Activity	Time in Phase To Date %	Estimated time (given Ep=635)
Planning	8%	51 minutes
Architecture	5%	32 minutes
Construction	36%	229 minutes
Refactoring	5%	32 minutes
Review	5%	32 minutes
Postmortem	6%	38 minutes
Sandbox	35%	222 minutes
Integration Test	0%	0 minutes
Total	100%	635 minutes

After calculating Ep, we can estimate how much time we are going to spend in each activity based on our historical data.