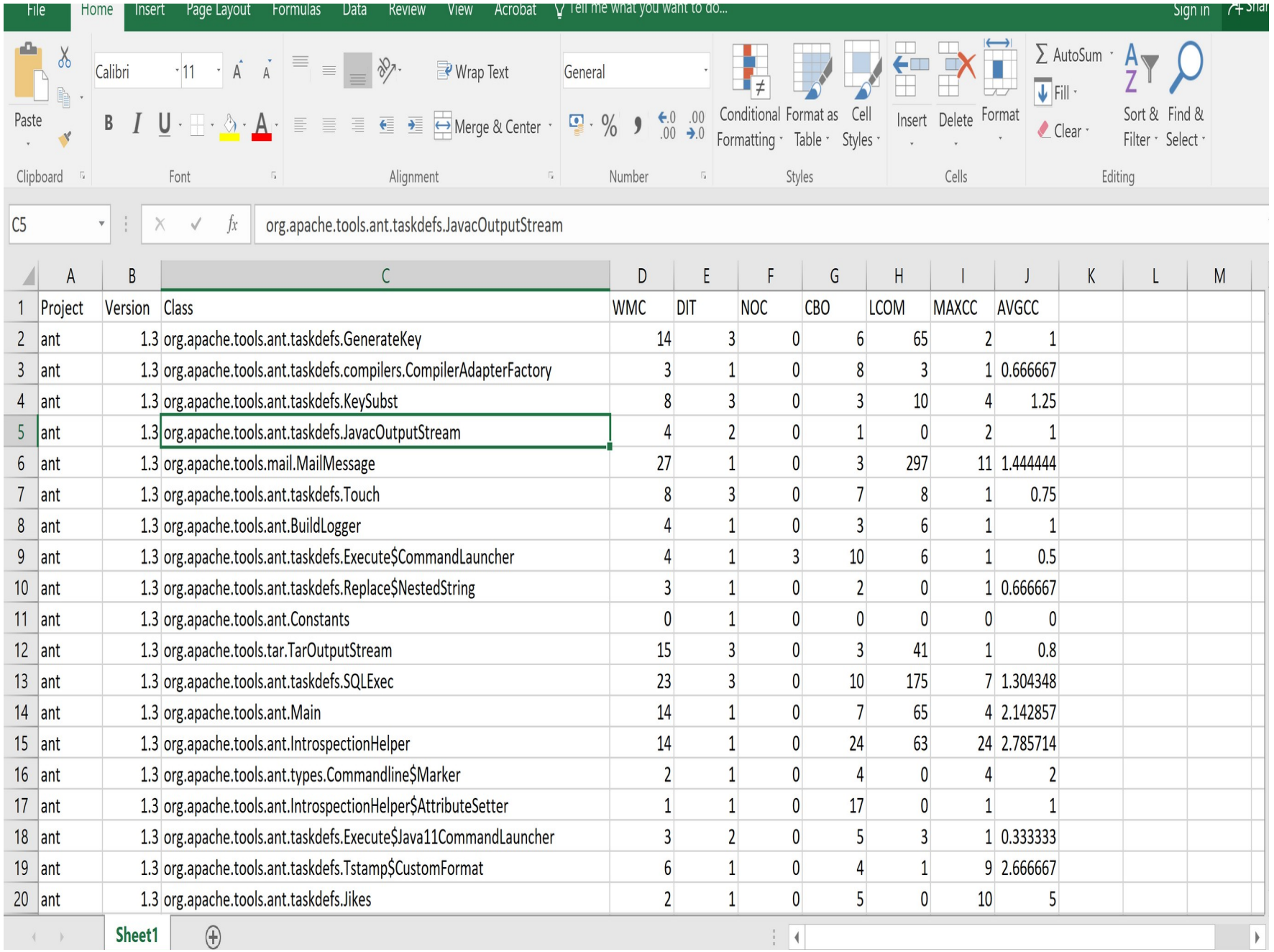
Software Engineering CS3003

Lecture 3: Software Metrics



File Home Insert Page Layout Formulas Data Review View Acrobat Tell me what you want to do... Sign in

Paste Calibri 11 Wrap Text General Conditional Formatting Format as Table Cell Styles Insert Delete Format AutoSum Fill Clear Sort & Filter Find & Select

Clipboard Font Alignment Number Styles Cells Editing

C5 org.apache.tools.ant.taskdefs.JavacOutputStream

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Project	Version	Class	WMC	DIT	NOC	CBO	LCOM	MAXCC	AVGCC			
2	ant	1.3	org.apache.tools.ant.taskdefs.GenerateKey	14	3	0	6	65	2	1			
3	ant	1.3	org.apache.tools.ant.taskdefs.compilers.CompilerAdapterFactory	3	1	0	8	3	1	0.666667			
4	ant	1.3	org.apache.tools.ant.taskdefs.KeySubst	8	3	0	3	10	4	1.25			
5	ant	1.3	org.apache.tools.ant.taskdefs.JavacOutputStream	4	2	0	1	0	2	1			
6	ant	1.3	org.apache.tools.mail.MailMessage	27	1	0	3	297	11	1.444444			
7	ant	1.3	org.apache.tools.ant.taskdefs.Touch	8	3	0	7	8	1	0.75			
8	ant	1.3	org.apache.tools.ant.BuildLogger	4	1	0	3	6	1	1			
9	ant	1.3	org.apache.tools.ant.taskdefs.Execute$CommandLauncher	4	1	3	10	6	1	0.5			
10	ant	1.3	org.apache.tools.ant.taskdefs.Replace$NestedString	3	1	0	2	0	1	0.666667			
11	ant	1.3	org.apache.tools.ant.Constants	0	1	0	0	0	0	0			
12	ant	1.3	org.apache.tools.tar.TarOutputStream	15	3	0	3	41	1	0.8			
13	ant	1.3	org.apache.tools.ant.taskdefs.SQLExec	23	3	0	10	175	7	1.304348			
14	ant	1.3	org.apache.tools.ant.Main	14	1	0	7	65	4	2.142857			
15	ant	1.3	org.apache.tools.ant.IntrospectionHelper	14	1	0	24	63	24	2.785714			
16	ant	1.3	org.apache.tools.ant.types.Commandline$Marker	2	1	0	4	0	4	2			
17	ant	1.3	org.apache.tools.ant.IntrospectionHelper$AttributeSetter	1	1	0	17	0	1	1			
18	ant	1.3	org.apache.tools.ant.taskdefs.Execute$Java11CommandLauncher	3	2	0	5	3	1	0.333333			
19	ant	1.3	org.apache.tools.ant.taskdefs.Tstamp$CustomFormat	6	1	0	4	1	9	2.666667			
20	ant	1.3	org.apache.tools.ant.taskdefs.Jikes	2	1	0	5	0	10	5			

Sheet1

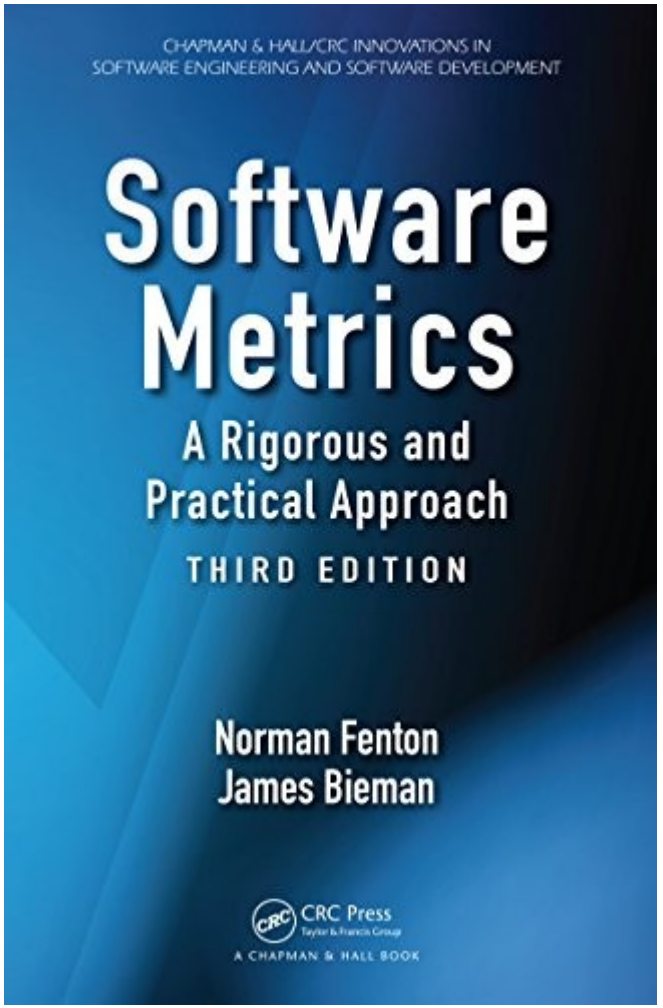
Lecture schedule

Week	Lecture Topic	Lecturer	Week Commencing
1	Introducing the module and Software Engineering	Steve Counsell	20th Sept.
2	Software maintenance and Evolution	Steve Counsell	27th Sept.
3	Software metrics	Steve Counsell	4th Oct.
4	Software structure, refactoring and code smells	Steve Counsell	11th Oct.
5	Test-driven development	Giuseppe Destefanis	18th Oct.
6	Software complexity **Coursework released Tues 26th Oct.**	Steve Counsell	25th Oct.
7	**ASK week**	**N/A**	**1st Nov**
8	Software fault-proneness	Steve Counsell	8th Nov.
9	Clean code	Steve Counsell	15th Nov.
10	Human factors in software engineering	Giuseppe Destefanis	22th Nov.
11	SE techniques applied in action	Steve Counsell	29th Dec.
12	Guest Lecture (tba) **Coursework hand-in 6th December**	Guest Lecture	6th Dec.

Lab schedule

Week	Labs	Week Commencing
1	**No labs**	20th Sept.
2	Lab (Introduction)	27th Sept.
3	Lab	4th Oct.
4	Lab	11th Oct.
5	Lab	18th Oct.
6	**No lab**	25th Oct.
7	**ASK week**	**1st Nov.**
8	Lab	8th Nov.
9	Catch-up Lab	15th Nov.
10	Work on coursework (no Lab)	22nd Nov.
11	Work on coursework (no Lab)	29th Nov.
12	**No lab**	6th Dec.

Norman Fenton



Structure of this lecture

This lecture will try to answer the following questions:

- How can software size be measured?
- How can software structure be measured?
- How can objected oriented code be measured?
- .....What is the link to the previous lecture on Maintenance and Evolution?

Uses of measurement

- Measurement helps us to “understand”
  - Makes the current activity visible
  - Measures can establish guidelines
    - Methods too long, classes too long etc.
- Measurement allows us to “control”
  - Predict outcomes and change processes
- Measurement encourages us to “improve”
  - When we measure our product we can establish quality targets and aim to improve

How can software size be measured?

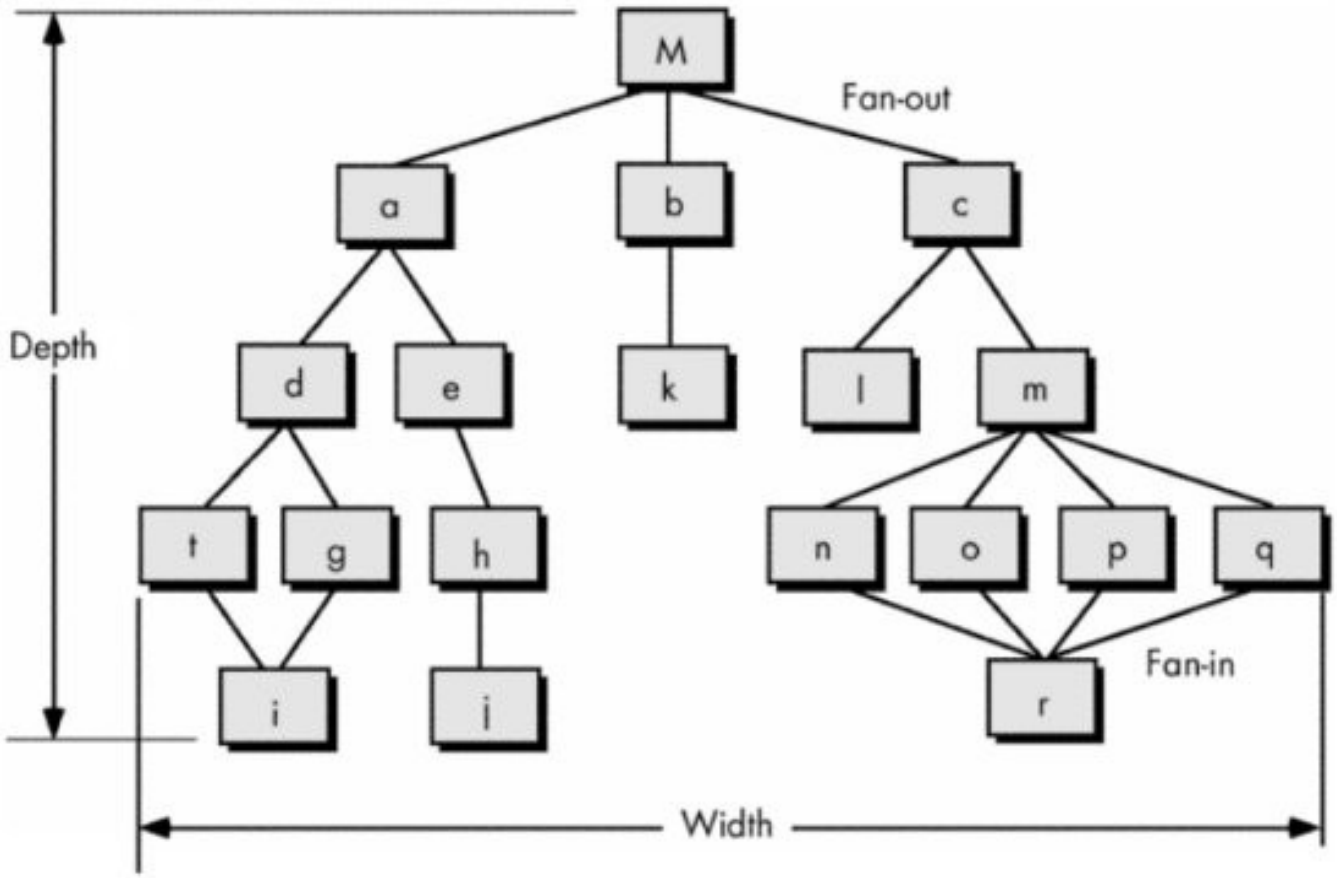
- Why is software size important?
  - Related to effort and cost
- Lines of code (LOC) is a common measure of size:
  - …but not very useful?
  - What about comment lines, blank space, "}"?
    - Are they all counted?
- Many companies measure functionality rather than code length

How can software structure be measured?

Information flow within the system

- Indicator of maintainability and “coupling”
- Identifies critical stress parts of the system and design problems
- Based on:
  - Fan-in: number of modules calling a module
  - Fan-out: number of modules called by a module

Fan-in and fan-out



Henry & Kafura's Complexity Metric:

- A module X is 10 lines long
- It has fan-in of 3 and fan-out of 2
- Complexity of a module = module length * $(\text{fan-in} * \text{fan-out})^2$

Complexity of X = $10 * (3 * 2)^2$

= 360

A complexity measure

- McCabe's Cyclometric Complexity measure
  - Commonly used by industry
  - In lots of tools
  - Based on control flow graph
  - Very useful for identifying white box test cases
- Attributed to Tom McCabe who worked for IBM in the 1970's

Cyclomatic Complexity (CC)

- CC(P) =
  #edges - #nodes +2

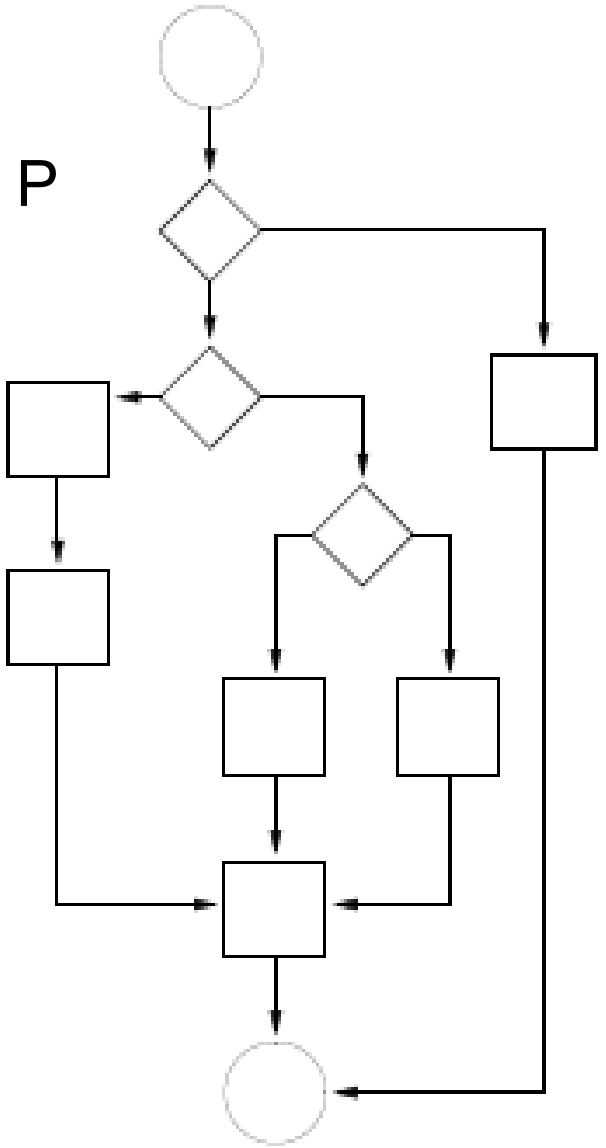
#edges = 11

#nodes = 9

CC = 4

(note: we exclude the start and end nodes and edges – although some sources don't)

Program P



```
stcode getlist(char *lin, int *i, stcode *status)
/*  1 */ {
/*  1 */   int num, done;
/*  1 */   line2 = 0;
/*  1 */   nlines = 0;
/*  1 */   done = (getone(lin,i,&num,status)!=OK);
/*  2 */   while (!done)
/*  3 */   {
/*  3 */     line1 = line2;
/*  3 */     line2 = num;
/*  3 */     nlines++;
/*  3 */     if (lin[*i]==SEMICOL)
/*  4 */       curln = num;
/*  5 */     if ((lin[*i]==COMMA)||(lin[*i]==SEMICOL))
/*  6 */     {
/*  6 */       *i = *i + 1;
/*  6 */       done = (getone(lin,i,&num,status)!=OK);
/*  6 */     }
/*  7 */      else
/*  7 */        done = 1;
/*  8 */   }
/*  9 */   nlines = min(nlines,2);
/*  9 */   if (nlines == 0)
/* 10 */     line2 = curln;
/* 11 */   if (nlines <= 1)
/* 12 */     line1 = line2;
/* 13 */   if (*status != ERR)
/* 14 */     *status = OK;
/* 15 */   return(*status);
/* 16 */ }
```

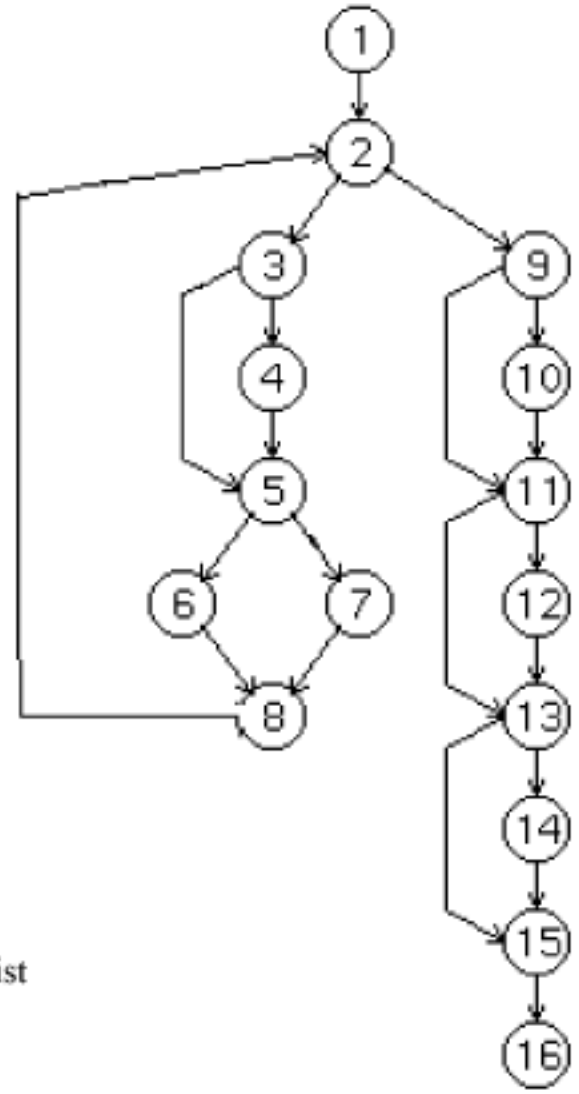


Figure 1. Program Getlist

Complexity Metrics

- Why use complexity metrics like CC?
  - They can be used to identify:
    - Candidate modules for code inspections
    - Areas where redesign may be appropriate
    - Areas where additional documentation is required
    - Areas where additional testing may be required
    - Areas for refactoring

Chidamber and Kemerer’s work has been referenced over 8000 times

IEEE TRANSACTIONS ON

SOFTWARE ENGINEERING

FEBRUARY 1985 VOLUME SE-11 NUMBER 2 (ISSN 0098-5589)

A PUBLICATION OF THE IEEE COMPUTER SOCIETY

How can OO code be measured?

- We'll look at 5 of the 6 metrics of Chidamber and Kemerer (C&K)
- Developed in 1991
  - Weighted methods per class (WMC)
    - A simple count of the number of methods in a class
  - Depth in the inheritance tree of a class (DIT)
    - The level in the inheritance tree of a class
  - Number of children (NOC)
    - The number of immediate sub-classes a class has

C&K metrics (cont.)

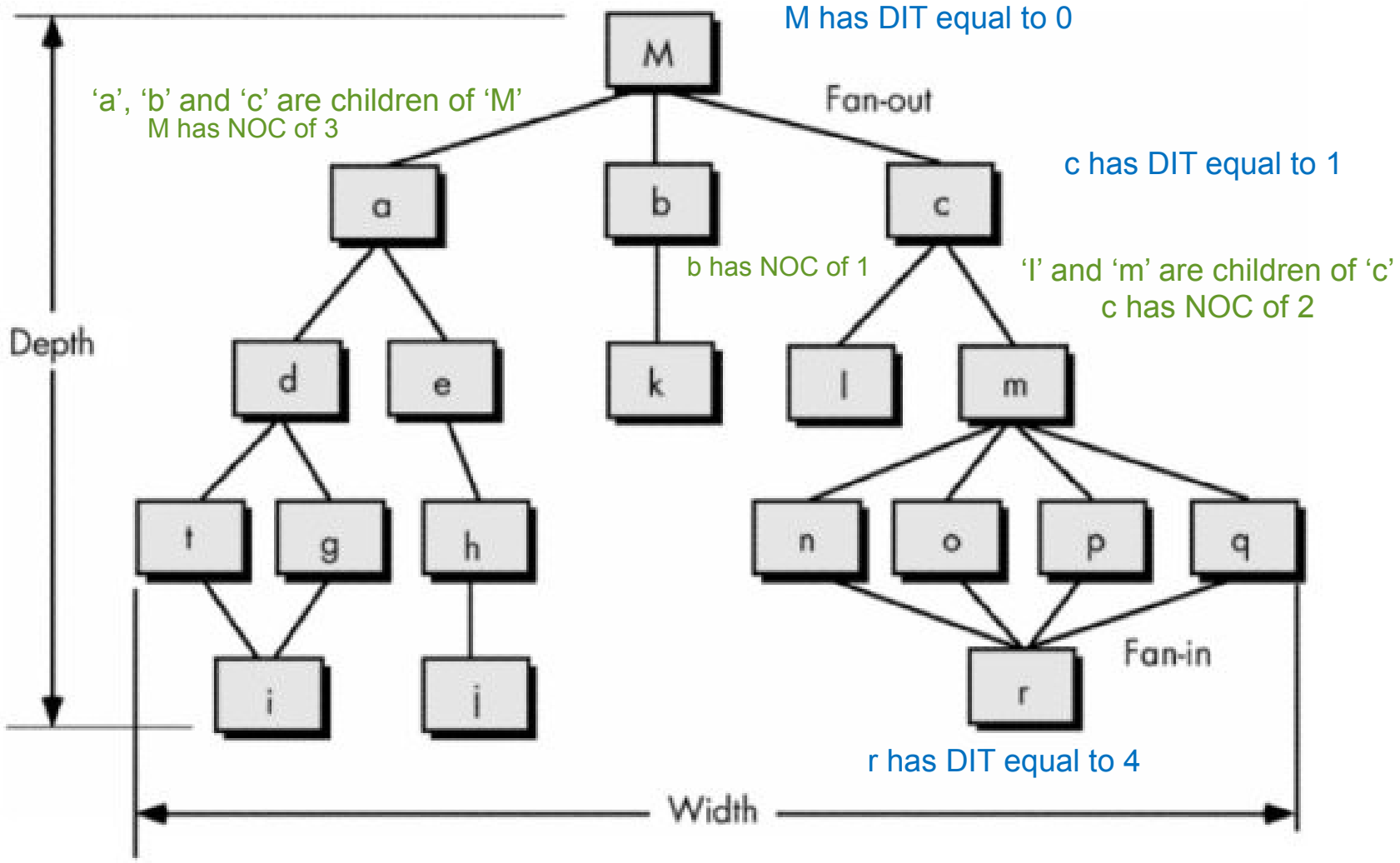
- Coupling between objects (CBO)
  - Number of other classes coupled to a class
- Lack of cohesion of methods in a class (LCOM)
  - Measure of how “well intra-connected” the fields and methods of a class are
  - The missing metric is called The Response for a Class
    - But nobody really understands what it is trying to measure

Weighted Methods per Class (WMC)

WMC

- The “weighted” part can be ignored
- Simply the number of methods in a class
- The number of methods is a predictor of how much time and effort is required to maintain the class
- Classes with large numbers of methods are likely to be more application specific, limiting the possibility of reuse

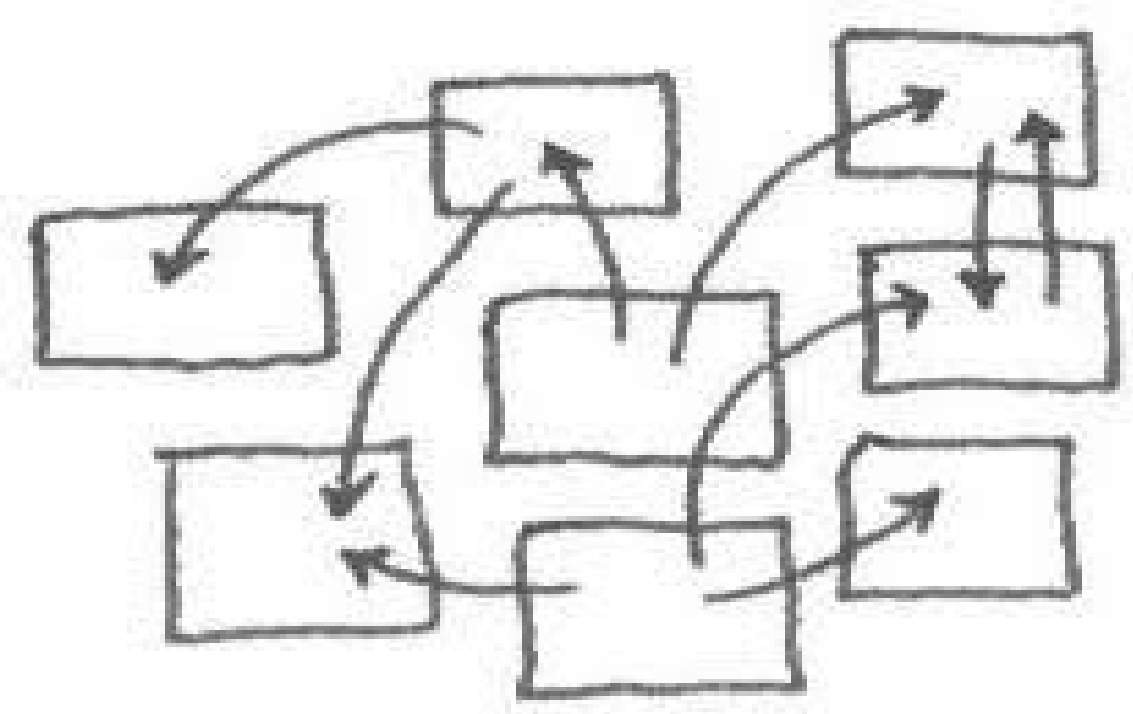
Depth in the Inheritance Tree of a Class (DIT) and Number of Children (NOC)



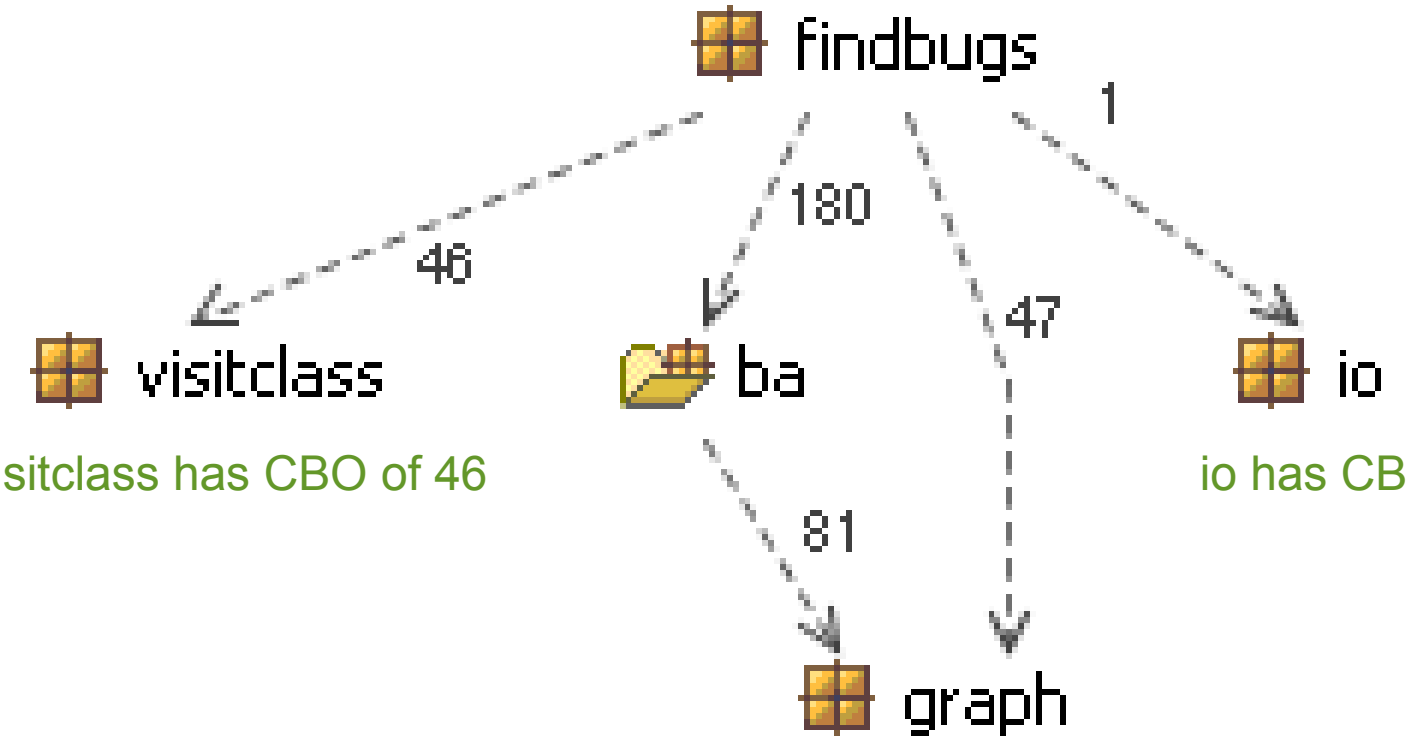
Ref: Marchese, PACE University

Coupling Between Objects (CBO)

CBO



Class coupling (ravioli code); here, boxes represent classes and arrows represent links (i.e., coupling) between classes



visitclass has CBO of 46

io has CBO of 1

graph has CBO of 81 + 47 = 128

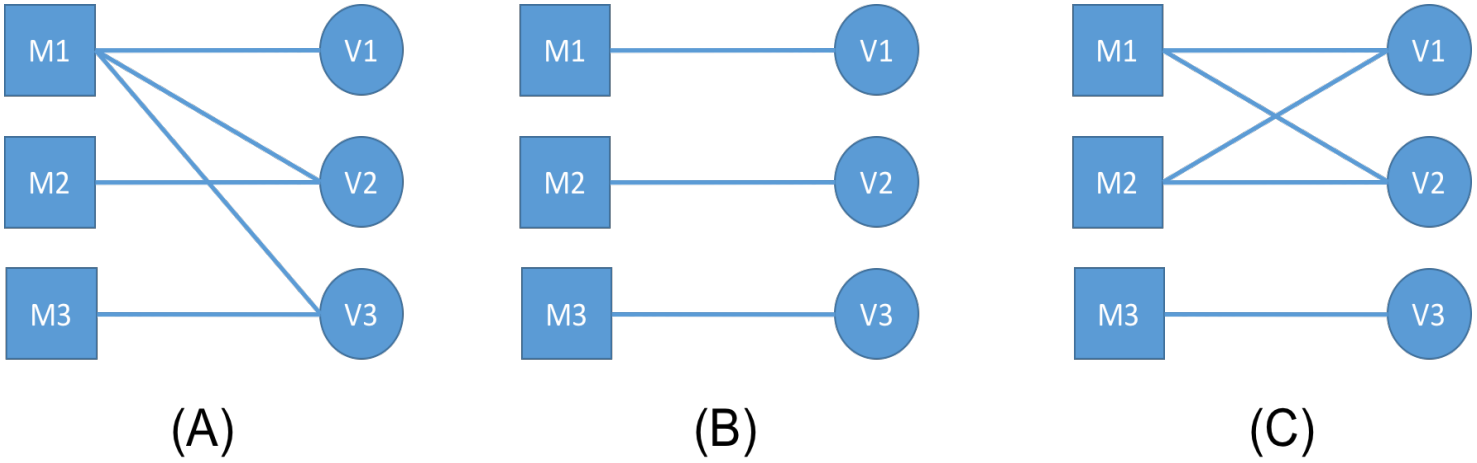
https://structure101.com/2008/11/27/software-erosion-findbugs/

Lack of cohesion of methods in a class (LCOM)

LCOM

- Cohesiveness of methods within a class is desirable, since it promotes encapsulation
- Lack of cohesion implies classes should probably be split into two or more subclasses
- Low cohesion increases complexity, thereby increasing the likelihood of errors during the development process
- (source: virtualmachinery.com/sidebar3.htm)

LCOM Example



Methods (M1-M3) and the variables those methods use (V1—V3)

http://www.tusharma.in/technical/revisiting-lcom/

Problems with metrics

- There is a tendency for professionals to display over-optimism and over-confidence in metrics
- Metrics may cause more harm then good
  - Data is shown because its easy to gather and display
- Metrics may have a negative effect on developer productivity and well-being
- Academic/industry divide
  - Industry may be interested in specific aspects of their code
    - For example, code churn, level of technical debt

Software Metric Usage

- Use common sense and organizational sensitivity when interpreting metrics data
- Provide regular feedback to the individuals and teams who have worked to collect measures and metrics
- Don’t use metrics to appraise individuals
- Never use metrics to threaten individuals or teams
- Metrics data that indicates a problem area should not be considered “negative”.
  - These data are merely an indicator for process improvement

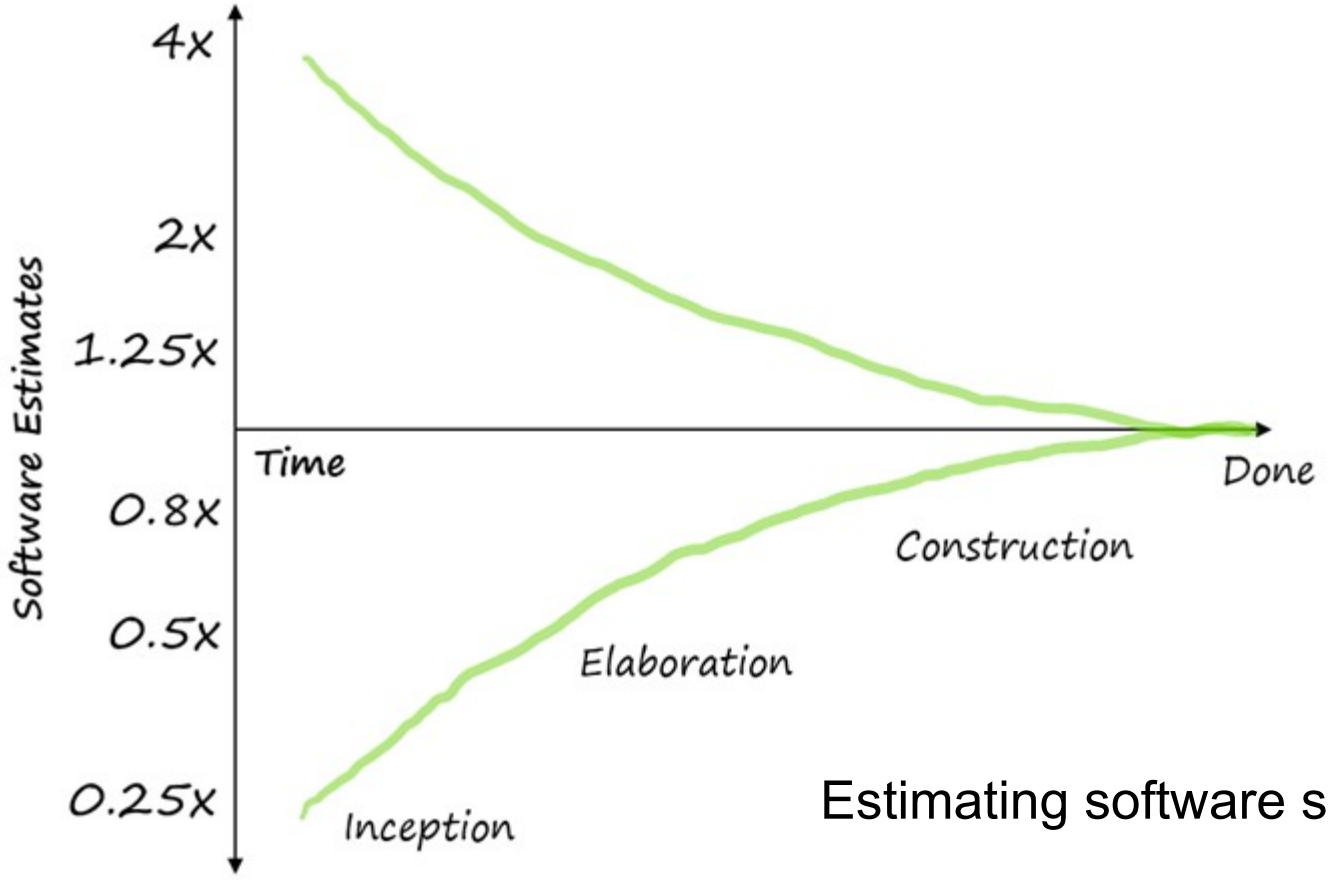
NASA's use of metrics

- What do NASA use on their code?
  - Cyclomatic Complexity
  - Lines of Code
  - Number of comments
  - Number of blank lines
  - Branch count
- NASA has one of the best metrics collection programmes that I know of:
  - The reason is simple

NASA metrics

Metrics	Description
BRANCH_COUNT	Branch count metric
LOC_TOTAL	Total number of source code lines
LOC_EXECUTABLE	Number of lines of executable code
LOC_COMMENTS	Number of lines of comments
LOC_BLANK	Number of blank lines
NUM_OPERATORS	Total number of operators
NUM_OPERANDS	Total number of operands
NUM_UNIQUE_OPERATORS	Number of unique operators
NUM_UNIQUE_OPERANDS	Number of unique operands
CYCLOMATIC_COMPLEXITY	Cyclomatic complexity of a module.
ESSENTIAL_COMPLEXITY	Essential complexity of a module.
DESIGN_COMPLEXITY	Design complexity of a module.
LOC_CODE_AND_COMMENT	Number of lines with both code and comments

Cone of uncertainty (McConnell)

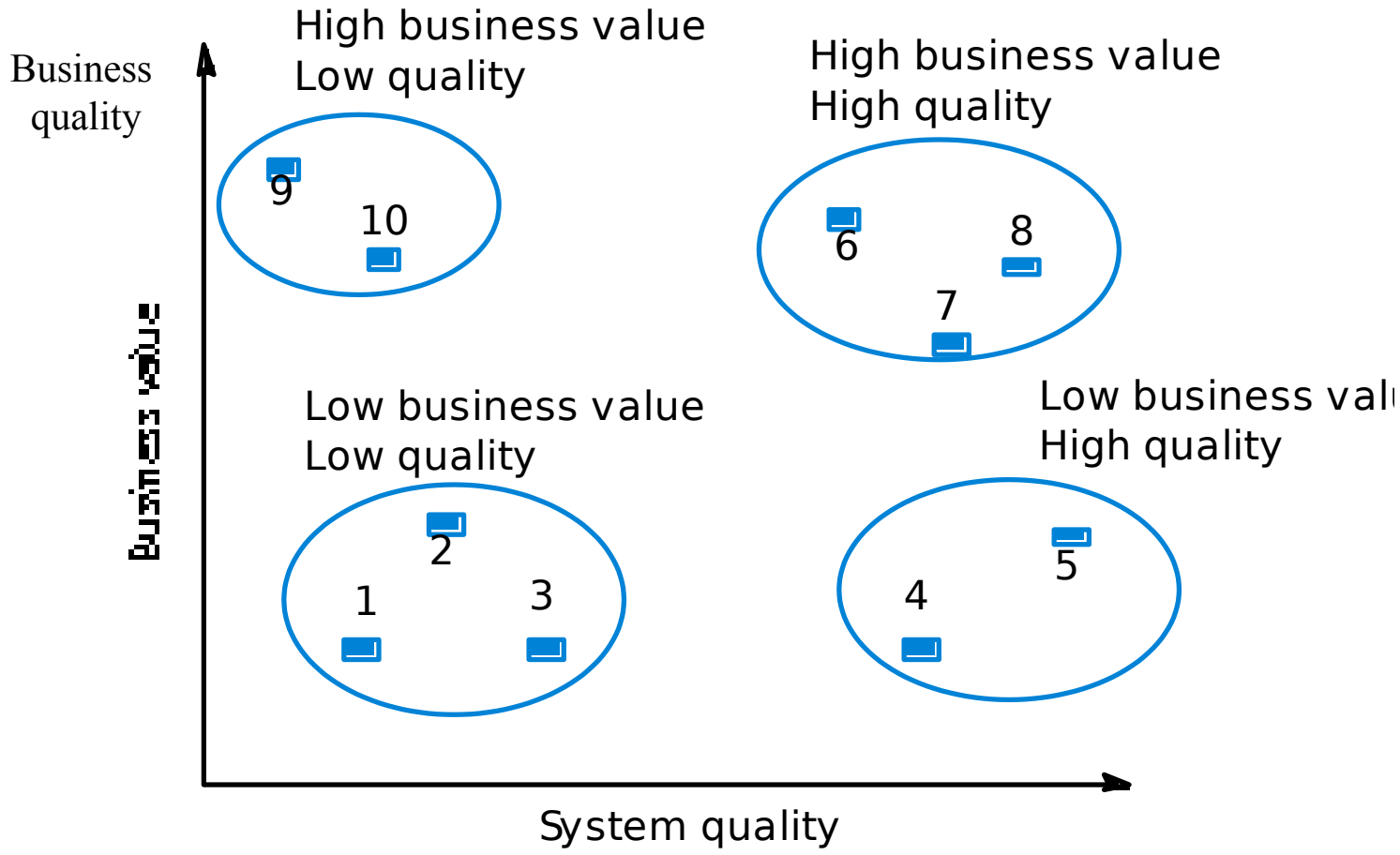


Estimating software size

What does the cone show?

- At the beginning of a software project:
  - Estimates are subject to large uncertainty
- As we progress, we learn more about the system and uncertainty decreases (estimates become more accurate)
- When we start writing code, that uncertainty decreases even more
  - The more features we complete, the more that uncertainty decreases
- Eventually, the uncertainty reaches 0% (at the project release)

How metrics can help us make decisions (an "audit grid")



Test-based metrics

- How many test cases have been designed per requirement?
- How many test cases have still to be designed?
- How many test cases have been executed?
- How many test cases passed or failed?
- How many bugs were identified and what were their severities?

Questions

- **Question 1:** What do you think is the value of using software metrics to measure code?
- **Question 2:** Discuss one disadvantage of any four of C&K's metrics of your choice.
- **Question 3:** "Small classes are less likely to be fault-prone than larger ones." To what extent do you agree with this statement?
- **Question 4:** Why are test-based metrics important?

Reading

- Sommerville – Ch. 24 in both editions 9 and 10
- Norman Fenton, James Bieman Software Metrics: A Rigorous and Practical Approach, Third Edition, CRC Press, 2014
- **Lines of Code (LOC)**
  - Why LOC is not good as a measure see: http://www.inf.u-szeged.hu/~beszedes/research/SED-TR2014-001-LOC.pdf

Reading (cont.)

- **Cyclomatic Complexity**
  - geeksforgeeks.org/cyclomatic-complexity/
- **C&K metrics**
- slideshare.net/skmetz/oo-metrics
- http://people.scs.carleton.ca/~jeanpier/sharedF14/T1/extra%20stuff/about%20metrics/Chidamber%20&%20Kemerer%20object-oriented%20metrics%20suite.pdf
- virtualmachinery.com/sidebar3.htm