More Fun With Product Metrics

Jamal Madni

CECS 445

Lecture 5: February 9th, 2021

The Plan This Week…

Step 1: Schedule Meeting With Your Customer ASAP to generate all “User Stories”

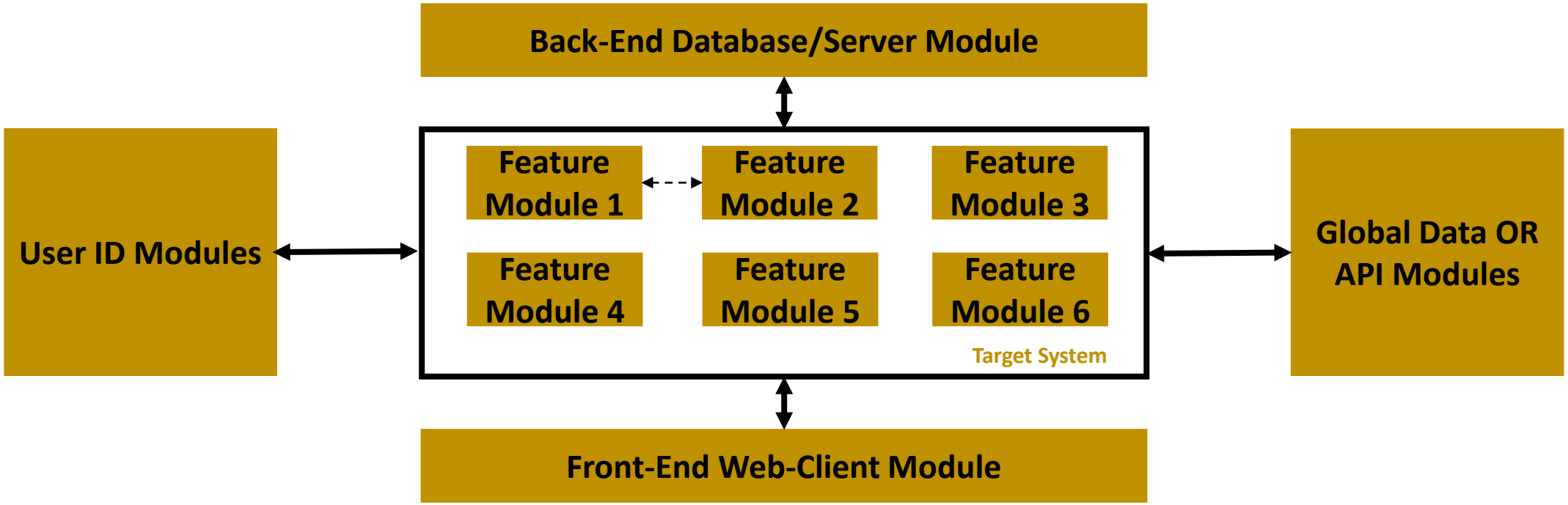
Title:	Priority:	Estimate:
User Story: As a [description of user], I want [functionality] so that [benefit].		
Acceptance Criteria: Given [how things begin] When [action taken] Then [outcome of taking action]		

Step 2: Complete Requirements Table *(25 – 30 Requirements)*

Software Requirement	Acceptance Criteria	Module	Module Function Point (FP) Score	Requirement Score	Priority	Lead	Time
"X shall perform Y"	"How do I know it's completed successfully?"	Specific Target System Module, Database Module, Front End Web-Client Module		FP, ABC or Halsted for Requirement	H/M/L		Number between 0 -1

The Plan This Week…

Step 3: Refine Architecture Diagram



Step 4: Which 5 requirements will you begin with for S1W2? Why?

Remember Key Metrics

Function Point (FP) Metrics

$$FP = \text{count total} \times [0.65 + 0.01 \times \Sigma(F_i)]$$

Information Domain Value	Count		Weighting factor: Simple	Average	Complex	
External Inputs (EIs)		3	3	4	6	=
External Outputs (EOs)		3	4	5	7	=
External Inquiries (EQs)		3	3	4	6	=
Internal Logical Files (ILFs)		3	7	10	15	=
External Interface Files (EIFs)		3	5	7	10	=
Count total						

External Inputs	*Parameters*
External Outputs	*Return Types, Error Messages, UI*
External Inquiries	*Function Calls By Others*
Internal Logical Files	*Local Data Dependent On Global Variables / Files*
External Interface Files	*Global States Used By Function*

1. Does the system require reliable backup and recovery?
2. Are specialized data communications required to transfer information to or from the application?
3. Are there distributed processing functions?
4. Is performance critical?
5. Will the system run in an existing, heavily utilized operational environment?
6. Does the system require online data entry?
7. Does the online data entry require the input transaction to be built over multiple screens or operations?
8. Are the ILFs updated online?
9. Are the inputs, outputs, files, or inquiries complex?
10. Is the internal processing complex?
11. Is the code designed to be reusable?
12. Are conversion and installation included in the design?
13. Is the system designed for multiple installations in different organizations?
14. Is the application designed to facilitate change and ease of use by the user?

Remember Key Metrics

Halsted Complexity

For a given problem, Let:

- η_1 = the number of distinct operators
- η_2 = the number of distinct operands
- N_1 = the total number of operators
- N_2 = the total number of operands

From these numbers, several measures can be calculated:

- Program vocabulary: $\eta = \eta_1 + \eta_2$
- Program length: $N = N_1 + N_2$
- Calculated estimated program length: $\hat{N} = \eta_1 \log_2 \eta_1 + \eta_2 \log_2 \eta_2$
- Volume: $V = N \times \log_2 \eta$
- Difficulty : $D = \frac{\eta_1}{2} \times \frac{N_2}{\eta_2}$
- Effort: $E = D \times V$

```
main()
{
    int a, b, c, avg;
    scanf("%d %d %d", &a, &b, &c);
    avg = (a + b + c) / 3;
    printf("avg = %d", avg);
}
```

The unique operators are: `main` , `()` , `{}` , `int` , `scanf` , `&` , `=` , `+` , `/` , `printf` , `,` , `;`

The unique operands are: `a` , `b` , `c` , `avg` , `"%d %d %d"` , `3` , `"avg = %d"`

- $\eta_1 = 12, \eta_2 = 7, \eta = 19$
- $N_1 = 27, N_2 = 15, N = 42$
- Calculated Estimated Program Length: $\hat{N} = 12 \times log_2 12 + 7 \times log_2 7 = 62.67$
- Volume: $V = 42 \times log_2 19 = 178.4$
- Difficulty: $D = \frac{12}{2} \times \frac{15}{7} = 12.85$
- Effort: $E = 12.85 \times 178.4 = 2292.44$
- Time required to program: $T = \frac{2292.44}{18} = 127.357$ seconds
- Number of delivered bugs: $B = \frac{2292.44^{\frac{2}{3}}}{3000} = 0.05$

Remember Key Metrics

ABC Software Metric

$$| < ABCvector > | = \sqrt{(A^2 + B^2 + C^2)}$$

The three components of the ABC score are defined as following:

- Assignment: storage or transfer of data into a variable.
- Branches: an explicit forward program branch out of scope.
- Conditionals: Boolean or logic test.

ABC rules for C++ [edit]

The following rules give the count of Assignments, Branches, Conditionals in the ABC metric for C++:

1. Add one to the assignment count when:
   - Occurrence of an assignment operator (exclude constant declarations and default parameter assignments) (**=, *=, /=, %=, +=, <<=, >>=, &=, !=, ^=**).
   - Occurrence of an increment or a decrement operator (prefix or postfix) (**++, --**).
   - Initialization of a variable or a nonconstant class member.
2. Add one to branch count when:
   - Occurrence of a function call or a class method call.
   - Occurrence of any goto statement which has a target at a deeper level of nesting than the level to the goto.
   - Occurrence of 'new' or 'delete' operators.
3. Add one to condition count when:
   - Occurrence of a conditional operator (**<, >, <=, >=, ==, !=**).
   - Occurrence of the following keywords ('**else**', '**case**', '**default**', '**?**', '**try**', '**catch**').
   - Occurrence of a unary conditional operator.

ABC rules for Java [edit]

The following rules give the count of Assignments, Branches, Conditionals in the ABC metric for Java:

1. Add one to the assignment count when:
   - Occurrence of an assignment operator (exclude constant declarations and default parameter assignments) (**=, *=, /=, %=, +=, <<=, >>=, &=, !=, ^=, >>>=**).
   - Occurrence of an increment or a decrement operator (prefix or postfix) (**++, --**).
2. Add one to branch count when
   - Occurrence of a function call or a class method call.
   - Occurrence of a 'new' operator.
3. Add one to condition count when:
   - Occurrence of a conditional operator (**<, >, <=, >=, ==, !=**).
   - Occurrence of the following keywords ('**else**', '**case**', '**default**', '**?**', '**try**', '**catch**').
   - Occurrence of a unary conditional operator.

Why Not These Metrics?

Project	LOC	Effort	$(000)	Pp. doc.	Errors	Defects	People
alpha	12,100	24	168	365	134	29	3
beta	27,200	62	440	1224	321	86	5
gamma	20,200	43	314	1050	256	64	6
...	...	...	...	...	...	...	...

- Errors per KLOC (thousand lines of code)
- Defects per KLOC
- $ per KLOC
- Pages of documentation per KLOC

In addition, other interesting metrics can be computed:

- Errors per person-month
- KLOC per person-month
- $ per page of documentation

Interesting Combinations?

Programming Language	LOC per Function Point			
	Avg.	Median	Low	High
Ada	154	–	104	205
ASP	56	50	32	106
Assembler	337	315	91	694
C	148	107	22	704
C++	59	53	20	178
C#	58	59	51	704
COBOL	80	78	8	400
ColdFusion	68	56	52	105
DBase IV	52	–	–	–
Easytrieve+	33	34	25	41
Focus	43	42	32	56
FORTRAN	90	118	35	–
FoxPro	32	35	25	35
HTML	43	42	35	53
Informix	42	31	24	57
J2EE	57	50	50	67
Java	55	53	9	214
JavaScript	54	55	45	63
JSP	59	–	–	–
Lotus Notes	23	21	15	46

Programming Language	LOC per Function Point			
	Avg.	Median	Low	High
Mantis	71	27	22	250
Natural	51	53	34	60
.NET	60	60	60	60
Oracle	42	29	12	217
OracleDev2K	35	30	23	100
PeopleSoft	37	32	34	40
Perl	57	57	45	60
PL/1	58	57	27	92
Powerbuilder	28	22	8	105
RPG II/III	61	49	24	155
SAS	50	35	33	49
Smalltalk	26	19	10	55
SQL	31	37	13	80
VBScript	38	37	29	50
Visual Basic	50	52	14	276

What do these tables mean? What implications for a developer metric?