Software Project Management

Lab 3

February 12, 2020

Anna Safonov 100601514
Amin Khakpour 100669547
Adam Wong Chew Onn 100598499

Table of Contents

Estimated Effort

Calculating estimated effort with the COCOMO model requires estimates on the KLOC (thousands of lines of code) for the project. COCOMO is a model based on a fitting regression formula using data from historical projects. It requires knowing the programming language and frameworks that will be used before the design and implementation phase, which my group can infer with some level of certainty based on our limited experience. It also requires an estimate of KLOC based on the language, the framework, and the historical similar projects. No reliable data were found for a project with similar architecture, languages, and frameworks to draw an estimated KLOC.

Thus, our group decided that it would be more accurate and reliable to base our estimated effort analysis on the Function Points (FPs) model and then use FPs and Language Factor (LF) tables to estimate KLOC.

Function Points is also a parametric model just like COCOMO, but it focuses on estimating system size based on five types of functionality common in information systems.

Function Points:

- External input (EI) - input transactions that modify internal system files;
- External Query (EQ) - user-initiated transactions, which provide information but do not modify internal system files;
- External Output (EO)- transactions that extract and display data from internal system files;
- Internal Logical File (ILF) - data store within the system for analysis;
- External Interface File (EIF) - data store external to the system, maintained by a different application.

Our group decided to work on the fall detector software for this project, and the infrastructure requirements we have elicited in the last lab report based on our research and preliminary design are the following:

- Application and database servers
- Data collection: fall sensors (height, velocity, orientation), speaker, mic, GPS, heartbeat sensor
- Docking station for the necklace - charging, malfunction scanning, firmware update
- The light-weight smooth necklace shell (decision on appropriate material still pending)
- Cellular connectivity (mobile provider infrastructure to sync data between the device and the server and to allow VoIP)
- Mobile application and web interface

These components can further be divided into software (mobile application and web interface, embedded software for the necklace and the charging station) and hardware elements (sensor, speaker, mic, GPS, servers WiFi chip). The focus of this effort estimation part of the project is on estimating effort for software components.

Embedded elements for this product will be written in C programming language, while the mobile application will be written in React Native to allow for seamless cross-platform function. React Native is a framework based on React.js, which is written in JavaScript.

The web app can also be written with React.js and Angular.js, both of which are JavaScript frameworks. The limited database queries for retrieval of patient information can be implemented effectively with SQL, and an SQL-based data management system, such as MySQL.

In summary, our product’s development environment will include three different programming languages - JavaScript, C, and SQL; and four frameworks - React Native, React.js, Angular.js, and MySQL.

Functional Point Analysis:

Embedded system for necklace device:

EI: (low, 1) - user input to turn off fall alarm (false alarm or problem resolved)

EI: (low, 1) - user input to turn on fall alarm (fall was not automatically registered, or patient needs emergency services)

EI: (medium, 1) - VoIP input over the microphone

EI: (medium, 1) - sensor data for the heartbeat

EI: (medium, 1) - sensor data for velocity

EI: (medium, 1) - sensor data for height

EI: (medium, 1) - sensor data for speed

EI: (medium, 1) - sensor data for the geographical location (GPS)

EO: (medium, 1) - soft alarm to indicate the device is charged but not on the user

EO: (low, 1) - VoIP output over the speaker

EO: (high, 1) cellular signal to emergency services to indicate fall has been detected and the patient requires clarification of the status

EO: (medium, 1) - internal malfunction detected

EO: (low, 1) - battery status

LIF: (medium, 4) - sensor status - 4 separate points for height, velocity, orientation, and heartbeat

LIF: (medium, 4) - sensor data - 4 separate points for height, velocity, orientation, and heartbeat

Embedded system for docking station:

EI: (low, 1) - device battery status

EI: (medium, 1) - device firmware version

EI: (medium, 1) - device function status

EO: (low, 1) - time to full charge

EO: (medium, 1) - status of firmware update

LIF: (medium, 1) - the latest firmware drivers stored

EIF: (high, 1) - server for updates, technical support, scanning tools, etc.

Mobile application:

EI: (medium, 1) - user login credentials

EO: (medium, 1) - user credential validation status

EO: (low, 1) - health statistics

EO: (low, 1) - emergency call statistics

EO: (medium, 1) - false-negative and false-positive alarms invoked by the patient’s device

EO: (low, 1) - battery usage statistics

EQ: (high, 1) - current patient status

EQ: (high, 1) - current patient location

LIF: (medium, 4) - sensor data - 4 points

LIF: (high, 1) - user profile data, including limited health information

EIF: (high, 3) - emergency services data/status/reports - 3 points

Web application:

(same functionality as the mobile app, but access through a browser)

EI: (medium, 1) - user login credentials

EO: (medium, 1) - user credential validation status

EO: (low, 1) - health statistics

EO: (low, 1) - emergency call statistics

EO: (medium, 1) - false-negative and false-positive alarms invoked by the patient’s device

EO: (low, 1) - battery usage statistics

EQ: (high, 1) - current patient status

EQ: (high, 1) - current patient location

LIF: (medium, 4) - sensor data - 4 points

LIF: (high, 1) - user profile data, including limited health information

EIF: (high, 3) - emergency services data/status/reports - 3 points

	Complexity						
	Low		Medium		High		
Description	**Estimate**	*****	**Estimate**	*****	**Estimate**	*****	**Total**
EO	9	4	7	5	1	7	78
EI	3	3	10	4		6	49
EQ		3		4	4	6	24
LIF		7	11	10	2	15	140
EIF		5		7	6	10	60
						Total Function Points (FP)	**351**
						Environmental Factor	**Rating**
						Data Communication	5
						Distributed Computing	4
						Performance Requirements	5
						Constrained Configuration	4
						Transaction Rate	5
						Online Inquiry and/or Entry	5
						End-User Efficiency	4
						Online Update	3
						Complex Processing	2
						Reusability	2
						Ease of Conversion/Install	5
						Ease of Operation	5
						Used at Multiple Sites	2
						Potential for Future Change	2
						Total Environmental Factor (N):	**53**
						Complexity Adjustment Factor (CAF):	**1.18**
						Adjusted Function Points (AFP):	**414.18**

Figure 1. Function Point Estimation for the overall project

The above calculations were done based on multiplier values specified on slide 43 lecture 5 material and function point example on slide 46 lecture 5.

CAF = 0.65 + (0.01*Total Environmental Factor)

CAF = 0.65 + (0.01*53) = 1.18

AFP = CAF*FP = 1.18*351 = 414.18

To estimate KLOC for this project, we need to separate the tasks done for different languages (C, JavaScript, and SQL), because each language has a corresponding language factor.

Based on the functional point analysis outlines above, the embedded components for this project will be written in C, the web application and mobile application will be written in JavaScript, and we assume that all queries through applications will be written in SQL. It is important to note here that these estimates will not be accurate, because with the use of frameworks, the number of lines of code that the developer has to write is significantly reduced through templates and libraries provided by the frameworks. Since we anticipate that the KLOC estimation will be higher then the actual KLOC, the scope of work for this software development process will be overestimated to a certain degree.

FP estimates for each subsystem based on programming language:

	Complexity						
	Low		**Medium**		**High**		
Description	**Estimate**	*****	**Estimate**	*****	**Estimate**	*****	**Total**
EO	3	4	3	5	1	7	34
EI	3	3	8	4		6	41
EQ		3		4		6	0
LIF		7	9	10	0	15	90
EIF		5		7	1	10	10
Total Function Points (FP)							**175**
Environmental Factor							**Rating**
Data Communication							5
Distributed Computing							4
Performance Requirements							5
Constrained Configuration							4
Transaction Rate							5
Online Inquiry and/or Entry							5
End-User Efficiency							4
Online Update							2

Complex Processing	1
Reusability	1
Ease of Conversion/Install	5
Ease of Operation	5
Used at Multiple Sites	1
Potential for Future Change	2
Total Environmental Factor (N):	**49**
Complexity Adjustment Factor (CAF):	**1.14**
Adjusted Function Points (AFP):	**199.5**

Figure 2. Function Point Estimation for the embedded software components (C language)

CAF = 0.65 + (0.01*Total Environmental Factor)

CAF = 0.65 + (0.01*53) = 1.14

AFP = CAF*FP = 1.14*175 = 199.5

	Complexity						
	Low		**Medium**		**High**		
Description	**Estimate**	*****	**Estimate**	*****	**Estimate**	*****	**Total**
EO	6	4	4	5		7	44
EI		3	2	4		6	8
EQ		3		4		6	0
LIF		7		10	2	15	30
EIF		5		7	6	10	60
						Total Function Points (FP)	**142**
						Environmental Factor	**Rating**
						Data Communication	5
						Distributed Computing	4
						Performance Requirements	5
						Constrained Configuration	4
						Transaction Rate	3
						Online Inquiry and/or Entry	5
						End-User Efficiency	4
						Online Update	3
						Complex Processing	2

Reusability	2
Ease of Conversion/Install	5
Ease of Operation	5
Used at Multiple Sites	2
Potential for Future Change	2
Total Environmental Factor (N):	**51**
Complexity Adjustment Factor (CAF):	**1.16**
Adjusted Function Points (AFP):	**164.72**

Figure 3. Function Point Estimation for the web and mobile software components (JavaScript)

CAF = 0.65 + (0.01*Total Environmental Factor)

CAF = 0.65 + (0.01*51) = 1.16

AFP = CAF*FP = 1.16*142 = 164.72

	Complexity						
	Low		**Medium**		**High**		
Description	**Estimate**	*****	**Estimate**	*****	**Estimate**	*****	**Total**
EO		4		5		7	0
EI		3		4		6	0
EQ		3		4	4	6	24
LIF		7		10		15	0
EIF		5		7		10	0
						Total Function Points (FP)	**24**
						Environmental Factor	**Rating**
						Data Communication	5
						Distributed Computing	3
						Performance Requirements	5
						Constrained Configuration	4
						Transaction Rate	5
						Online Inquiry and/or Entry	5
						End-User Efficiency	5
						Online Update	5
						Complex Processing	3

Reusability	2
Ease of Conversion/Install	5
Ease of Operation	5
Used at Multiple Sites	2
Potential for Future Change	2
Total Environmental Factor (N):	**56**
Complexity Adjustment Factor (CAF):	**1.21**
Adjusted Function Points (AFP):	**29.04**

Figure 4. Function Point Estimation for the database components (SQL)

CAF = 0.65 + (0.01*Total Environmental Factor)

CAF = 0.65 + (0.01*56) = 1.21

AFP = CAF*FP = 1.21*24 = 29.04

We anticipate that the estimates for language-grouped subcomponents are more accurate than the overall estimates.

According to the data provided by Quantitative Software Management (QSM), average language factor (LF) for C is 97, for JavaScript is 47, and for SQL is 21. Based on these data, the estimated KLOC for each subsystem is:

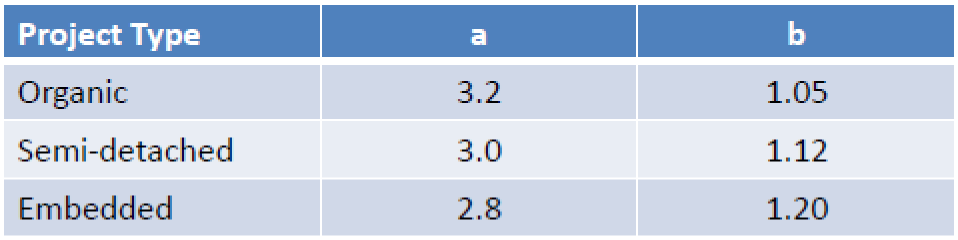
$SLOC_{embedded} = FP_{embedded} * LF_{embedded} = 199.5 * 97 = 19\ 351.5$ KLOC

$SLOC_{app} = FP_{app} * LF_{app} = 164.72 * 47 = 7\ 741.84$ KLOC

$SLOC_{db} = FP_{db} * LF_{db} = 29.04 * 21 = 609.84$ KLOC

$SLOC_{overall} = 19\ 351.5 + 7\ 741.84 + 609.84 = 27\ 703.18$ LOC or 28 KLOC

Based on the definitions of different types of projects provided in lecture 5 slide 28, we classify this project as embedded system project, as none of our team members have experience with developing a multi-language system that involves both hardware and software and stringent requirements for patient information privacy and safety, and responding to emergency situations in which a patient's life may be in danger.



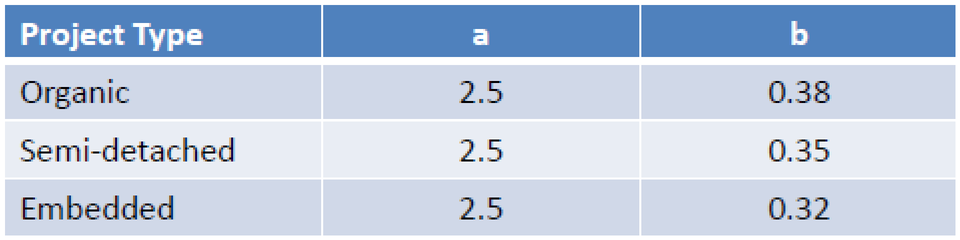
Project Type	a	b
Organic	3.2	1.05
Semi-detached	3.0	1.12
Embedded	2.8	1.20

Figure 5. CoCoMo constants for different types of projects (slide 29, lecture 5)

Therefore, our effort estimate for overall project is:

$E = 2.8 * KLOC_{overall}^{1.20} = 2.8 * (28)^{1.20} = 152.67 = 153$ person-months

We can further break down the timeline by calculating the estimated effort per each subsystem, which may be done later in the project. The duration is then estimated based on the following modifiers:



Project Type	a	b
Organic	2.5	0.38
Semi-detached	2.5	0.35
Embedded	2.5	0.32

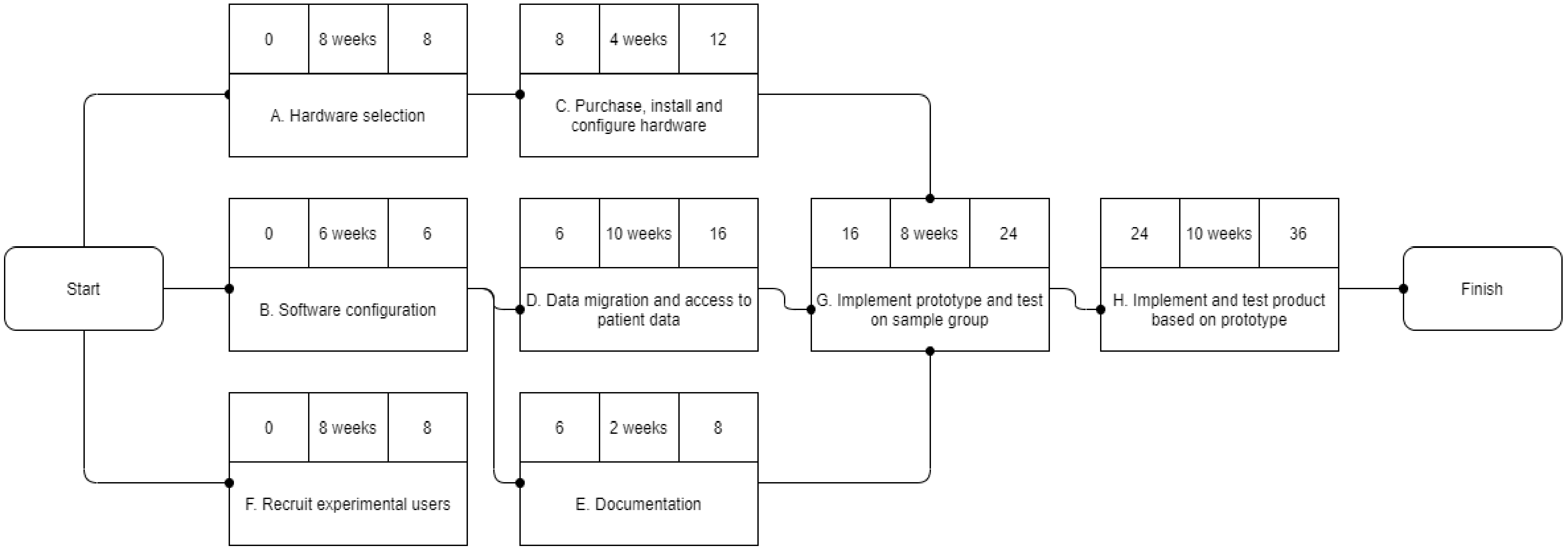
Figure 6. CoCoMo Duration Estimation, slide 30, lecture 5

$D = 2.5 * E^{0.32} = 2.5 * (153)^{0.32} = 12.5$ months

Activity Diagram

12.5 months estimated duration = 54 weeks

A. **Hardware selection**: research and make decisions regarding application and DB servers (specs, etc.) required to support the functionality of the web and mobile applications and the user devices; research and make decisions regarding the material and other hardware components for the device and the docking station.
   Duration: 8 weeks
B. **Software configuration:** Since languages and frameworks have already been picked, purchase any licenses install and configure software languages, IDEs, testing frameworks, etc.
   Duration: 6 weeks
C. **Purchase, install and configure hardware**
   Duration: 4 weeks
D. **Data migration and access to patient data:** obtain HIPPA, set up DB for device syncing, set up and configure data pipe for querying patient information
   Duration: 10 weeks
E. **Documentation:** document the development process, software capabilities and functionality, testing, etc. Write user guides.
   Duration: 2 weeks
F. **Recruit experimental users:** recruit users (elderly patients in a hospital or community setting) to test out prototype device
   Duration: 8 weeks
G. **Implement prototype and test on sample group:** incorporate any changes, improvements, troubleshooting discovered and requested during prototype testing.
   Duration: 8 weeks
H. **Implement and test product based on prototype:** use insight learned from building and testing prototype to build the product and test it.
   Duration: 10 weeks



Start
0
8 weeks
8
A. Hardware selection
8
4 weeks
12
C. Purchase, install and configure hardware
0
6 weeks
6
B. Software configuration
6
10 weeks
16
D. Data migration and access to patient data
16
8 weeks
24
G. Implement prototype and test on sample group
24
10 weeks
36
H. Implement and test product based on prototype
Finish
0
8 weeks
8
F. Recruit experimental users
6
2 weeks
8
E. Documentation

Risks and Counter-measures

Risk	**Risk reduction techniques**
Personnel shortfalls	Implement rigorous hiring process, hire staff with skills and experience relevant to the project and technologies used in the project. Implement team building exercises and lunch and learn sessions.
Unrealistic time and cost estimates	Use multiple estimation techniques and update estimates as the project progresses. Standardize the development processes and other business processes in the project.
Developing the wrong software and hardware functions	Implement multiple user surveys along the process of development, test the prototype extensively with target group, write detailed documentation both for developers and for users.
Developing the wrong user interface	Prototype early and test with multiple target user groups.
Gold plating	Use prototype feedback to implement only the necessary requirements
Late changes to requirements	Use incremental development process; implement and train the team on Agile process.
Shortfalls in externally supplied components	Use benchmarking, inspections, formal specifications, contractual agreements, quality controls.
Shortfalls in externally performed tasks	Select vendors based on competitive bids for high quality work.
Real time performance problems	Use prototype to ensure real-time performance quality; test final product – acceptance testing.
Development technically too difficult	Implement detailed technical analysis, cost-benefit analysis, prototyping, train staff and hire competent staff with required qualifications and experience.