Software Estimation – Complexity or Density?

Murali Chemuturi & Sarada Kaligotla

Most of the software size estimation techniques, be it Function Point Analysis or Use Case Point Analysis, adjust the size of software using “Complexity Factors” – the names may differ, but it remains one or two Complexity Factors.

I often felt that “Complexity” is a misnomer in this scenario. Let me present you my views.

The paradox of Complexity Vis-à-vis Size

"Everything is simpler than you think and at the same time more complex than you imagine." - (Johann Wolfgang von Goethe)

What is complexity? There are no accepted definitions of Complexity either !!

I quote Dr. Sam Vaknin `(http://samvak.tripod.com)`, from his paper “The Complexity of Simplicity” –

- A straight line can be described with three symbols (A, B, and the distance between them) - or with three billion symbols (a subset of the discrete points which make up the line and their inter-relatedness, their function). But whatever the number of symbols we choose to employ, however complex our level of description, it has nothing to do with the straight line or with its "real world" traits. The straight line is not rendered more (or less) complex or orderly by our choice of level of (meta) description and language elements.

This implies that it is up to us to make something complex or simple by the choice of words or symbols.

These are few of the interpretations of the word “Complexity” –

1. We do not fully comprehend something – hence we say that it is “Complex”
2. There are unforeseen “hurdles” on the way and we are not sure how many hurdles and how to get across – so we may say that it is “Complex”
3. The work is “delicate”, that is, if we are not careful something may break or fail or cause severe damage – therefore, we label it as “Complex”
4. There may not be room for maneuvering/manipulation – then again, we may label it as “Complex”

Often we label some thing as complex, if we do not understand or lack the capability to either understand or do it. For example,

- For a householder, plumbing is a complex activity while it is a simple task for a professional plumber
- For an orthopedic surgeon who can construct a joint in the human body, carpentry (which is far simpler task than operating bones and joints) becomes complex
- For a carpenter plumbing is complex, for a plumber electrification work is complex, for an electrician masonry work is complex
- Extending the above argument, COBOL is complex for a Java programmer, GUI (Graphical user Interface) programming is complex for a CUI (Character User Interface) programmer, Java is complex for a VB Programmer and so on

I am sure that you got the drift by now – I am implying that we find an activity complex when we are not trained in it or are naturally handicapped to do it.

It is also a fact that we do not allow a person to do a job in which he is not adept (skilled) at. A novice is never allowed to handle tasks independently unless he is closely supervised by a senior artisan. Since we do not assign a task to an unskilled person, is it proper to put a measure of complexity to the software development? Surely we don't allow a COBOL programmer to write Java programs and vice-versa !!

Therefore, is it proper to label a component (Function Point, Object Point or Use Case etc.) as simple, average or high in complexity? What we can surely say is that a component is small, medium or large sized components.

Here I come to the next incongruity on this aspect – does complexity come in three convenient steps in the nature?

Complexity is a continuum – it is analog in nature. It cannot be restricted to three levels only. By putting complexity in three levels, people, including experts, can have different perceptions especially in borderline cases.

If I may say so, what is wrong by saying the complexity level is 2.5 or 3.75 or 5.25 and so on?

Another incongruity is adjusting the size owing to the complexity. Consider this –

A: "How much distance did you walk to day?"
B: "Today I walked on an even surfaced side walk – so it would be three miles"
A: "Why did you mention the quality of surface (complexity) you walked on? What possible bearing it has on the answer to the question I asked?"
B: "See I adjust the distance by the type of surface on which I walk. Yesterday, I walked indoors on my walking machine – thus it would be 3.9 miles. The day before, I walked on a mountain trail, it would be six miles"
A: "Wow – how did you arrive at these figures?"
B: "I walk for 45 minutes. I adjust the distance I walked based on the quality of surface (complexity) I walked on. On an even surfaced sidewalk, the multiplication factor is 1 – indoors on a walking machine, the multiplication factor is 1.3 and on a mountain trail the multiplication factor is 2"

Would you silently accept the explanation given by B? Not likely. You are wont to say that the distance does not change due to the quality of surface (complexity) but the rate of walking (productivity) changes.

Let us consider another scenario.

A: "How many steps (on a staircase in a multi storied complex) are there?"
B: "That depends on the climber"
A: "What do you mean?"
B: "I will explain. We adjust the count of the stairs (points) based on the climber (Programmer) capability"
"If the age of the person is between 3 to 5 years, we multiply the count of stairs by a factor 10"
"If the age of the person is between 6 and 12, then we multiply the count of stairs by a factor of 1.2"

"If the age of the person is between 13 and 18, then we multiply the count of stairs by a factor of 0.9"
"If the age of the person is between 19 and 35, then we multiply the count of stairs by a factor of 1.0"
"If the age of the person is between 36 and 45, then we multiply the count of stairs by a factor of 1.35"
If the age of the person is between 46 and 55, then we multiply the count of stairs by a factor of 3
If the age of the person is between 56 and 65, then we multiply the count of stairs by a factor of 5
If the age of the person is between 66 and above, then we multiply the count of stairs by a factor of 20

Would you agree with the explanation given by B?

You are likely to tell him "Look Mister, you should know that the count of stairs does not change – they always remain the same. What changes is the rate-of-climbing (productivity) the stairs. Where did you learn that size changes because of the person's capability?"

Thus you can see that the size remains constant irrespective of the complexity and personnel expertise. What changes is the **rate** (**productivity**) at which the act is accomplished (productivity).

So when complexity increases, productivity decreases – but not the size!

When personnel expertise is low, the productivity is low – but the size remains the same.

In both the cases, the size does not change !! Remember that Mount Everest remains at an altitude of twenty nine thousand feet (approximately or 8,848 meters / 29,028 feet to be precise) irrespective whoever tries to climb it !!

Then why do all present software estimation techniques adjust the size due to complexity or programmer capability or similar factors?

Density – not Complexity

What term perhaps, should we use that increases the size, in my opinion, is "**Density**".

Same volume of liquid with more density is heavier than a liquid of lesser density.

Same volume of solid too, with more density is heavier than a solid of lesser density – a good example is the question we ask youngsters – "which is heavier – a dime made of gold or steel?" to trick them into saying dime made of steel!!

In engineering, drawings come in normally five sizes –
1. A0 – measuring one Square Metre
2. A1 – measuring half Square Metre
3. A2 – measuring one-fourth Square Metre
4. A3 – measuring one-eighth Square Metre
5. A4 – measuring one-sixteenth of Square Metre

You ask any Engineering Draftsman as to how much time it will take to prepare a A0 size drawing – he will tell you that it depends on the “Density” of the lines/pictures that are packed on to the drawing!!

Here we have a parallel for our software development.

A screen, even when it is developed in the same development platform, takes different amount of time depending on the density of controls we place on the screen!!

A report similarly, takes different amount of time based on the density of data packed into it.

Take an example close to computers.

An integrated chip that has more transistors packed into it takes more time to develop than an IC that has lesser number of transistors. The chipmakers use the term “Density” and not complexity to describe their chips. They perhaps, say, that they are now packing a million transistors per square inch as against one hundred thousand earlier.

Thus, a functionality achieved with lesser number of lines code has lesser density that is achieved with more number of lines – this is assuming that the minimum possible LOC are used in both cases.

As can be seen, I think and propose that we use “Density” in place of Complexity when it comes to software estimation.

****

About the Authors:

Murali Chemuturi is a Fellow of Indian Institution of Industrial Engineering and a Senior Member of Computer society of India. He is a veteran of software development industry and is presently leading Chemuturi Consultants, which provides software products for software estimation and software project management.

Sarada Kaligotla has completed her Master’s in Computer Applications and is a certified PMP (PMI) and CSQA (QAI). She presently works for Blue Cross Blue Shield of MA. She has around 6 years experience in software industry with development and project management experience.

We appreciate your feedback – please feel free to mail your feedback murali@chemuturi.com

****