Software Development Time Estimation Using Similarity Theory

Alexander Dymo. 23 August 2005

Why should we use another theory?

Currently we know about many ways to estimate software development time and cost and still the estimation in general is considered to be "not that scientific" (Shafer). Why does that happen? That's because estimation is in fact (COCOMO, COCOMO II) performed using regression models which only approximate the influence of various parameters on software development time and cost. Obviously, those approximations do not take the nature of software development into consideration.

From the other side, it looks (and it is) impossible to know the exact impact of those numerous (Fairley counts 10 structural parameters that consist of large number of sub parameters) parameters on the resulting time/cost. Therefore it becomes evident that no strict mathematical model is possible to create.

So, what kind of model would be appropriate for estimation? My experience in studying heat exchange processes tells that the model we need is called "physical model" which is usually not strict but provides good results because it takes into account physical nature of process. Software development is also a process; the only difference is that it's not physical so a physical model could not be built. But we know that physical model is a subtype of "analog model" and we can build that model of software development process. The only question is how.

Once again, from heat exchange we know about Similarity theory and namely one method of building analogous models which is called "method of units analysis" (explained best by Guhman). What would we get if we use that method and build the model of software development process? Let's take a look at that.

Similarity theory and method of units analysis

Similarity theory says that it's possible to generalize the experimental data taken from one process and apply generic relationships to know the behaviour of another, similar process.

Two processes are considered to be similar when:

- two processes belong to one class of processes (the class would define a "type" of software being developed – GUI library, IDE, media player, whatever else);
- they have geometrical similarity (always true for software because it has only one dimension which defines size be it SLOC or FP or else unit);
- they have time similarity (for software development this means that lifecycles should be similar, i.e. have the same type – spiral, waterfall, etc.);
- they have similarity of environment constants (economical factors like mean developer salary or technical factors like available tools and/or programming languages);
- initial and boundary conditions are also similar for two processes (software evolves only in time so no boundary conditions are appropriate, but initial conditions should be similar).

In the definitions above A "similar" to B means that $A=kB$ where k is a number.

Similarity theory says that it's possible to:

1) gather experimental data from one process – software time/cost and any parameters that influence the time/cost;
2) build a model that establish a relationships between parameters and time/cost in the form of equation:

$$R = A \cdot P_1^a \cdot P_2^b \cdot \ldots \cdot P_n^z \qquad (1)$$

where R – criterion which includes parameters to be defined (time/cost),
P_n – criterions which group parameters that influence on R,
A, a, b, $\ldots z$ – numerical constants defined experimentally for each class of similar processes;

3) calculate numerical constants using one set of experimental data (gathered, for example, during development of one version of some product);

4) apply the model equation for other similar processes: substitute $P_1 - P_n$ with new values and calculate estimated R.

Method of units analysis tells us how to build an analog model (equation (1)).

The process of applying the method is:

1. Define a set of determinative parameters which influence on the process.
2. Build a system of measurement units for those parameters.
3. Define a relationship between parameters in the form of equation

$$a = f(a_1, a_2, \ldots, a_k, a_{k+1}, \ldots, a_n) \qquad (2)$$

where a – a determinate parameter (time/cost),

a_1, a_2, …, a_n – determinative parameters which influence on a,

n – the number of parameters that can be measured,

k – the number of independent measurement units;

4. Find a relationship between non-dimensional criterions using π-theorem:

$$\pi = f_1(\pi_1, \pi_2, \ldots, \pi_{n-k}), \qquad (3)$$

where π, π_1, π_2, …, π_{n-k} – criterions.

5. Calculate numerical constants of the model.

An example of software cost estimation for software development process

At this moment we will consider development of imaginary software product using generic (read waterfall) lifecycle model and try to build a model that can be applied when estimating time/cost of similar products.

1. Defining a set of determinative parameters which influence on the process.
2. Building a system of units for those parameters.

Parameter	***Name***	***Unit***
Time of development	***t***	day
Size	***sz***	FP (points)
Complexity (Shafer)	***dcx***	FP / (man·day)
Requirement complexity (the number of necessary, steady functional requirements) (Shafer)	***rc***	func

Parameter	*Name*	*Unit*
Reuse volume (the specific cover of functional requirements by ready components) (Shafer)	***rv***	comp / func
Number of components	***cc***	comp
Number of developers	***dc***	man
Developer productivity (Albrecht)	***dp***	FP / (man·day)
Language factor	***lf***	FP / day
Developer experience (Curtice)	***dxp***	bug / (man·day)
Quality (Shaffer)	***q***	bug / FP

3. Defining a relationship between parameters in the form of equation

$$t = A \cdot dcx^a \cdot rc^b \cdot rv^c \cdot sz^d \cdot dc^e \cdot dp^f \cdot q^g \cdot lf^h \cdot dxp^i \cdot cc^j. \qquad (4)$$

4. Finding a relationship between non-dimensional criterions using π-theorem

Let's substitute parameters with their units:

$$day = \frac{fp^a}{man^a \cdot day^a} \cdot func^b \cdot \frac{comp^c}{func^c} \cdot fp^d \cdot man^e \times$$
$$\times \frac{fp^f}{man^f \cdot day^f} \cdot \frac{bug^g}{fp^g} \cdot \frac{fp^h}{day^h} \cdot \frac{bug^i}{man^i \cdot day^i} \cdot comp^j.$$

Number of determinative parameters: $n = 10$; number of independent units: $k = 6$; number of π – criterions to get $n - k = 4$.

Let's write a system of equations that contain exponents for each unit:

$$\left.\begin{array}{ll} \texttt{day:} & 1 = -a - f - h - i \\ \texttt{fp:} & 0 = a + d + f - g + h \\ \texttt{man:} & 0 = -a + e - f - i \\ \texttt{comp:} & 0 = c + j \\ \texttt{bug:} & 0 = g + i \\ \texttt{func:} & 0 = b - c \end{array}\right\}.$$

If we solve the system, we'll get:

$$\left.\begin{array}{l} g = -i \\ c = -j \\ b = -j \\ a = e - f - i \\ d = 1 \\ h = -1 - e \end{array}\right\}.$$

Let's substitute those values in the equation (4):

$$t = A \cdot \frac{dcx^{e}}{dcx^{f} \cdot dcx^{i}} \cdot \frac{1}{rc^{j}} \cdot \frac{1}{rv^{j}} \cdot sz \cdot dc^{e} \cdot dp^{f} \cdot \frac{1}{q^{i}} \cdot \frac{1}{lf \cdot lf^{e}} \cdot dxp^{i} \cdot cc^{j}.$$

Let's group parameters with the same exponents:

$$\frac{t \cdot lf}{sz} = A \cdot \left(\frac{dcx \cdot dc}{lf}\right)^{e} \cdot \left(\frac{dp}{dcx}\right)^{f} \cdot \left(\frac{cc}{rc \cdot rv}\right)^{j} \cdot \left(\frac{dxp}{q \cdot dcx}\right)^{m}.$$

Those groups of parameters are our non-dimensional criterions:

criterion of time and size:

$$Dts = \frac{t \cdot lf}{sz};\ [Dts] = \frac{day}{fp} \cdot \frac{fp}{day} = 1.$$

criterion of complexity:

$$Dlcom = \frac{dcx \cdot dc}{lf};\ [Dlcom] = \frac{fp}{man \cdot day} \cdot man \cdot \frac{day}{fp} = 1;$$

criterion of productivity:

$$Dprod = \frac{dp}{dcx};\ [Dprod] = \frac{fp}{man \cdot day} \cdot \frac{man \cdot day}{fp} = 1;$$

criterion of reusability:

$$Dreus = \frac{cc}{rc \cdot rv};\ [Dreus] = \mathrm{comp} \cdot \frac{1}{\mathrm{func}} \frac{\mathrm{func}}{\mathrm{comp}} = 1;$$

criterion of quality:

$$Dqua = \frac{dxp}{q \cdot dcx};\ [Dqua] = \frac{\mathrm{bug}}{\mathrm{man} \cdot \mathrm{day}} \cdot \frac{\mathrm{fp}}{\mathrm{bug}} \cdot \frac{\mathrm{man} \cdot \mathrm{day}}{\mathrm{fp}} = 1.$$

The resulting model equation in this case is:

$$Dts = A \cdot Dlcom^{e} \cdot Dprod^{f} \cdot Dreus^{j} \cdot Dqua^{i}. \qquad (5)$$

6. Calculating numerical constants of the model.

If we had real (and not imaginary) project completed, we would be able to calculate exponents e, f, i, j and A. After that we would be able to use this model equation (5) for estimation of development time of other similar software products.

Conclusions

1. For each instance of software development process it is easy to acquire analog models of the process.
2. Such models would make software time/cost estimation extremely easy for each similar software development process.
3. Each software developer (company, person, etc.) can use the method explained above to get their own customized models which would reflect their processes better (the set of parameters can be completely different from the example above).
4. Model quality depends only on the set of parameters. If parameters were chosen correctly, the model will be correct. There's no kind of approximation necessary.

Afterword

I realize that the text above is strictly theoretical and was not proved in real life. The example model equation shown above will probably not work as good as possible for custom processes. I expect some day I will be able to try it out and get some real life data and real life models to prove that similarity theory is also applicable for software development.

Literature

[Guhman] Гухман А.А. Физические основы теплопередачи. Том первый. Теория подобия и ее приложения. – М.: Государственное энергетическое издательство, 1934. – 316с.
(Guhman A.A. Physical basics of heat transfer. Volume 1. Similarity theory and its appliances)

[Shaffer] Шафер Д.Ф., Фартрелл Р.Т., Шафер Л.И. Управление программными проектами: достижение оптимального качества при минимуме затрат. – М.: Издательский дом “Вильямс”, 2003. – 1136 с.
(Shafer D.F. , Furtrell R.T., Shafer L.I. Quality Software Project Management)

[Fairley] Fairley R.E. "Recent Advances in Software Estimation Techniques." Proceedings of the 14th international IEEE software engineering conference, NY: IEEE Computer Society Press, pp.382-391.

[Albrecht] Albrecht A.J. Measuring Application Developers Productivity. Proceedings of the IBM Application Development Symposium, October 1979, CA. pp. 83-92.