2-2019

# Optimal Distribution of Testing Resources Between Different System Levels

Griselda Acosta
*University of Texas at El Paso*, gvacosta@miners.utep.edu

Eric Smith
*University of Texas at El Paso*, esmith2@utep.edu

Vladik Kreinovich
*University of Texas at El Paso*, vladik@utep.edu

Comments:
Technical Report: UTEP-CS-19-06

# Optimal Distribution of Testing Resources Between Different System Levels

Griselda Acosta[1], Eric Smith[2], and Vladik Kreinovich[3]
[1]Department of Electrical and Computer Engineering
[2]Department of Industrial, Manufacturing, and Systems Engineering
[3]Department of Computer Science
University of Texas at El Paso
500 W. University
El Paso, Texas 79968, USA
gvacosta@miners.utep.edu, esmith2@utep.edu, vladik@utep.edu

**Abstract**

When designing a system, we need to perform testing and checking on all levels of the system hierarchy, from the most general system level to the most detailed level. Our resources are limited, so we need to find the best way to allocate these resources, i.e., we need to decide how much efforts to use of each of the levels. In this paper, we formulate this problem in precise terms, and provide a solution to the resulting optimization problem.

## 1 Formulation of the Problem

**Need for system design.** Sometimes, engineers and scientists concentrate on designing a specific device or a specific software. However, no device and no software works on its own, whatever we design will be a part of a system. For example, when we design a new industrial plant, we need to take into account how its functioning will affect the natural ecosystem, how the increased transportation will affect the city infrastructure, how the new people brought to this plant will change the demographic system, etc. Similarly, in science, when we design a new radiotelescope (or even software to process signals from the radiotelescope), we need to take into account that this telescope will be mostly used as a part of a system of radiotelescopes and other astronomical instruments to observe different celestial objects.

**System design is hierarchical.** To properly design a system, it is important to first have a clear general structure. After that, once it becomes clear what are the system components and how they supposed to interact, we can move to designing these individual components – taking into account the need for these components to efficiently work together. These components usually also are subsystems, so we need to come up with their own components, etc. At the end, once all the tasks have been clarified, we proceed to the most detailed level, where we design individual machines and instruments and write the corresponding software.

Of course, the above sequence is an idealized representation of the actual design process: sometimes, after we go to a more detailed level of design, we realize the need to make some changes in the previously decided higher-level design structure. However, most of the time, the system design follows the above hierarchical pattern.

**Need for testing and checking.** On each design level, we need to check for possible problems and flaws. Flaws can occur on different levels.

For example, on the highest general-system level, we may forget an important aspect of the system – e.g., when designing a plant, we may not think about its ecological impact – and as a result, once the design is done, it may have to be redone completely. To avoid such situations, it is important to check the design on each level before starting a more detailed design level.

Flaws may also occur on the very lowest most-detailed level: e.g., we can have a software that does not always provide the correct control for the plant.

**Need to allocate testing resources between different levels.** We need to perform testing and checking on different levels of the system hierarchy. However, our testing resources are limited. It is therefore important to efficiently distributed the available resources between different levels; see, e.g., [2, 3, 5, 6, 7, 8, 9, 10, 11, 12].

**What we do in this paper.** In this paper, we describe the problem of allocating resources in precise terms, and we provide a solution to the resulting optimization problem.

## 2 Analysis of the Problem

**The cost of errors on different levels.** Errors can occur on all the levels:

- we can make an error on the highest level, by deciding on a faulty overall design;
- we can also make an error on the most detailed level, e.g., making an error when manufacturing one of the system's components.

An error on a higher level is very costly: if there was indeed an error in the overall design, we have to redo the overall design and thus, redo all the details – i.e., largely, start "from scratch". On the other hand, errors on the lower levels are not that costly: if we erred in designing one small component, then all we need to do is re-design this small component.

Let us number the levels from the most general one – which will be Level 1, via the next-detailed Level 2, then even-more-detailed Level 3, etc., all the way to the most detailed Level. Let us denote the overall number of levels by $n$. Then, the most detailed level is Level $n$.

In general, an error on each level $i$ leads to the need of redoing several details on the next-detailed level $i+1$. Let us denote the average number of details that need to be redone by $q$. Then, an error on Level $i$ necessitates redoing $q$ details on the next-detailed Level $i+1$. Each of these re-doings requires redoing $q$ details on the next level $i+2$; thus, an error on Level $i$ requires re-doing $q^2$ details on Level $i+2$. Similarly, we conclude that it requires re-doing $q^3$ details on Level $i+3$, and, in general, $q^k$ details on level $i+k$. In particular, for $k=n-i$, we conclude that an error on Level $i$ requires redoing $q^{n-i}$ details on the most detailed Level $n$.

Let $c$ denote the average cost of redoing a single detail on the most-detailed Level $n$. Then, the overall cost of an error on Level $i$ can be obtained by multiplying this per-error cost $c$ by the total number of details $q^{n-i}$ that need to be corrected, and is, thus, equal to $c \cdot q^{n-i}$.

**The cost of discovering errors.** How does the number $N$ of remaining errors depend on the effort – i.e., equivalently, on the time $t$ spent to find these errors. We would like to find a general formula $N(t)$ for describing this dependence.

It is important to take into account that there are different way to count errors. For example, when we talk about software errors, we can count the number of modules that do not perform as we intended, it we can count the number of lines of code where we made a mistake, or we can count the number of erroneous operations on each line of code. All three (and other) ways of counting errors make sense – but they differ by a factor. For example, to go from the number of erroneous moduli to the number of erroneous lines of code, we need to multiply the number of erroneous moduli by the average number of erroneous lines of code in an erroneous modulus. Thus, if we change the way we count errors, we go from the original number $N(t)$ to the new number $C \cdot N(t)$, where $C$ is the corresponding factor.

Both the original function $N(t)$ and the new function $C \cdot N(t)$ make sense. Thus, instead of a single function $N(t)$ for describing how the number of remaining errors depends on time $t$, we should consider the whole family of functions $\{C \cdot N(t)\}_C$ corresponding to all possible value $C > 0$.

The time $t$ is the time from the moment when we started testing. This may sound well-defined, but in practice, it changes from one person to another. Some programmers try to run the very first version of the program that they wrote – and thus, start debugging the code right away. Other programmers first try some on-paper tests and only start running when they are reasonably sure that they eliminate the most obvious bugs. While the results of both programmers may be similar, the starting time for measuring $t$ is different for the second programmer: what happened for the first programmer at time $t$, for the second programmer, happens at time $t - t_0$, where $t_0$ is the time the second programmer spend analyzing his/her code before running it. This value $t_0$ may be different for different programmers. It is therefore reasonable to require that the approximating family $\{C \cdot N(t)\}_C$ should not change if we simply change the way we measure the time, i.e., if we go from $t$ to $t - t_0$.

In other words, the family $\{C \cdot N(t - t_0)\}_C$ corresponding to the shifted time $t - t_0$ should coincide with the original family $\{C \cdot N(t)\}_C$. This means, in particular, that for every $t_0$, the function $N(t - t_0)$ from the shifted family should belong to the original family, i.e., it should have a form

$$N(t - t_0) = C(t_0) \cdot N(t), \tag{1}$$

for some value $C(t_0)$ depending on $t_0$.

The function $N(t)$ describing the number of remaining errors after time $t$, this function is (non-strictly) decreasing: when $t < t'$, then we should have $N(t) \geq N(t')$. Thus, it is measurable, and therefore, the function $C(t_0) = N(t - t_0)/N(t)$ is also measurable, as the ratio of two measurable functions. It is known (see, e.g., [1]) that for measurable functions, the only solutions to equation (1) have the form $N(t) = N_0 \cdot \exp(-a \cdot t)$ for some coefficients $N_0$ and $a$; see [4].

Now, we are ready to formulate the problem in precise terms.

## 3 Formulation of the Problem in Precise Terms

**Towards the formulation.** We want to divide the overall people-time $T$ that we have allocated for testing into times $t_1, \ldots, t_n$ allocated to testing on different levels:

$$t_1 + \ldots + t_n = T. \tag{2}$$

According to the above formulas, for each level $i$, after the testing, we will have $N_0 \cdot \exp(-a \cdot t_i)$ errors. The cost of each error on this level is $c \cdot q^{n-i}$, so the overall cost of all these errors is $c \cdot q^{n-i} \cdot N_0 \cdot \exp(-a \cdot t_i)$.

The overall cost $E$ coming from all the remaining errors can be computed by adding the costs corresponding to different levels:

$$E = \sum_{i=1}^{n} c \cdot q^{n-i} \cdot N_0 \cdot \exp(-a \cdot t_i). \tag{3}$$

**Resulting formulation.** We want to select the times $t_1, \ldots, t_n$ – under the constraint (1) – so as to minimize the overall cost $E$ of all the errors.

In other words, we want to minimize the expression (2) under the constraint (1).

## 4 Solving the Resulting Optimization Problem

**Solving the problem.** A usual way to solve a constraint optimization problem is to use Lagrange multipliers, i.e., to reduce the original problem of minimizing a function $f(x)$ under a constraint $g(x) = 0$ to the unconstrained problem of minimizing an expression $f(x) + \lambda \cdot g(x)$, where the parameter $\lambda$ (known as *Lagrange multiplier*) has to be determined from the condition $g(x) = 0$.

In our case, the constraint has the form

$$\sum_{i=1}^{n} t_i - T = 0,$$

so the corresponding unconstrained optimization problem means minimizing the expression

$$\sum_{i=1}^{n} c \cdot q^{n-i} \cdot N_0 \cdot \exp(-a \cdot t_i) + \lambda \cdot \left(\sum_{i=1}^{n} t_i - T\right).$$

To find the minimum of this expression, we differentiate it with respect to each unknown $t_i$ and equate the resulting (partial) derivative to 0. As a result, we get the following formula:

$$c \cdot q^{n-i} \cdot N_0 \cdot (-a) \cdot \exp(-a \cdot t_i) + \lambda = 0,$$

i.e.,

$$\exp(-a \cdot t_i) = \frac{\lambda}{a \cdot c \cdot N_0} \cdot q^{n-i}.$$

Taking logarithms of both sides and dividing the result by $-a$, we get

$$t_i = (n - i) \cdot \frac{|\ln(q)|}{a} + c_1,$$

where we denoted

$$c_1 \overset{\text{def}}{=} -\frac{1}{a} \cdot \ln\left(\frac{\lambda}{a \cdot c \cdot N_0}\right).$$

Combining terms not depending on $i$ into a single expression, we get

$$t_i = c_2 - i \cdot \frac{|\ln(q)|}{a}, \tag{4}$$

where

$$c_2 \overset{\text{def}}{=} c_1 + n \cdot \frac{|\ln(q)|}{a}.$$

In line with the main idea of the Lagrange multiplier technique, to find the value $c_2$, we substitute the expression (4) into the constraint (1). As a result, we get

$$T = \sum_{i=1}^{n} t_i = n \cdot c_2 - \left(\sum_{i=1}^{n} i\right) \cdot \frac{|\ln(q)|}{a}.$$

Here,

$$\sum_{i=1}^{n} i = 1 + 2 + \ldots + n = \frac{n \cdot (n+1)}{2},$$

thus

$$T = n \cdot c_2 - \frac{n \cdot (n+1)}{2} \cdot \frac{|\ln(q)|}{a},$$

and so,

$$c_2 = \frac{T}{n} + \frac{n+1}{2} \cdot \frac{|\ln(q)|}{a}.$$

Thus, we arrive at the following formula.

**Resulting solution.** In situation where an error on the next level costs $q$ times less than the error on the previous level, and the number of detected errors decreases with detection time as $\exp(-a \cdot t)$, the optimal allocation of the overall testing time $T$ into times $t_1, \ldots, t_n$ allocated to each level has the form

$$t_i = \left( \frac{T}{n} + \frac{n+1}{2} \cdot \frac{|\ln(q)|}{a} \right) - i \cdot \frac{|\ln(q)|}{a}. \tag{5}$$

**Discussion.** In other words, the allocated time linearly decreases as we go from the most abstract level to the more and more detailed levels. The fact that we allocate most of the testing time to the highest level makes perfect sense: as we have mentioned, errors on this level are the costliest ones. That the decrease should be linear follows from the specific formulas of our model.

## Acknowledgments

This work was partially supported by the US National Science Foundation via grant HRD-1242122 (Cyber-ShARE Center of Excellence).

## References

[1] J. Aczél and J. Dhombres, *Functional Equations in Several Variables*, Cambridge University Press, 2008.

[2] T. Bahill, R. Botta, and J. Daniels, "The Zachman framework populated with baseball models", *Journal of Enterprise Architecture*, 2006, Vol. 2, No. 4, pp. 50–68.

[3] B. Dantu and E. Smith, "Diagnostic modeling for medical decision making", In: T. Doolen and E. Van Aken (eds.), *Proceedings of the 2011 Industrial Engineering Research Conference*, Reno, Nevada, May 21–25, 2011.

[4] V. Kreinovich, Th. Swenson, and A. Elentukh, "Interval approach to testing software", *Interval Computations*, 1994, No. 2, pp. 90–109.

[5] S. Ferreira, R. Valerdi, N. Medvidovic, J. Hess, I. Deonandan, T. Mikaelian, and G. Shull, "Unmanned and autonomous systems of systems test and evaluation: challenges and opportunities", *Proceedings of the 5th IEEE Systems Conference*, San Diego, California, April 5–8, 2010.

[6] R. Gona and E. Smith, "Healthcare enterprise quality assessment with a Zachman-Bayesian framework", In: T. Doolen and E. Van Aken (eds.), *Proceedings of the 2011 Industrial Engineering Research Conference*, Reno, Nevada, May 21–25, 2011.

[7] J. Hess, G. Agarwal, K. Cowart, I. Deonandan, C. R. Kenley, T. Mikaelian, and R. Valerdi, "Normative and descriptive models for test & evaluation of unmanned and autonomous systems of systems", *Proceedings of the 20th INCOSE Symposium*, Chicago, Illinois, July 12–15, 2010, Vol. 1, pp. 644–654.

[8] J. Hess and R. Valerdi, "Test and evaluation of a SoS using a prescriptive and adaptive testing framework", *Proceedings of the 5th IEEE International Conference on Systems of Systems Engineering*, Loughborough, UK, June 22–24, 2010.

[9] J. M. Stecklein, J. Dabney, B. Dick, B. Haskins, R. Lovell, and G. Moroney, "Error cost escalation through the project life cycle", *Proceedings of the 14th Annual International Symposium of the International Council on Systems Engineering (INCOSE)*, Toulouse, France, June 19–24, 2004; also available as NASA Johnson Space Center Technical Report JSC-CN-8435, https://ntrs.nasa.gov/search.jsp?R=20100036670.

[10] J. A. Zachman, "A framework for information systems architecture", *IBM Systems Journal*, 1999, Vol. 38, No. 2–3, pp. 454–470.

[11] *Zachman Framework*, https://en.wikipedia.org/wiki/Zachman_Framework, downloaded on January 29, 2019

[12] Francisco Zapata, Amanda Posadas, Aditya Akundi, and Eric Smith, "Combinatorial black box testing for genetic algorithms with discrete and continuous variables and potential applications for general testing methods", *Proceedings of the Joint Workshops on "System-of-Systems: A Network Integration Evaluation (NIE) Experience" and "R&D 'Use' of Propulsion Vehicles"*, El Paso, Texas, July 16–18, 2013.