Turtles All The Way Up: the Complexity and Hyperbolic Nature of Procedures

Andrew Frederick Cowie
Procedural

Abstract

There is a fundamental structure when considering the organization of work. The procedure to accomplish a task can be decomposed into a sequence of sub-steps, and likewise steps can be composed together to form more complex procedures and processes. The ability to do this arises because the unit of procedural work is fractal in nature and thus the same regardless of scale. This leads us to formally describe the step as the fundamental unit of procedure. We define the measure of complexity inherent in any such procedure, and propose a scientific notation for describing the magnitude and scale of this complexity. Finally we consider the relation between the complexity of a procedure and the complexity inherent in the design of a system. The paper ends with an analysis of the space that procedures occupy, showing it to be hyperbolic.

> *Unlike every other creature on the Discworld, the giant star turtle knew exactly where it was going.*
>
> Terry Pratchett

Introduction

Using a rigid structure to articulate the procedures needed to organize the activities of a small team was highly successful in helping software and systems engineers safely execute mission critical events. By defining a consistent hierarchical style to be used when documenting the instructions, the people running these events were able to combine concurrent activity with clearly defined coordination points.

In the original formulation [2], each step had one or more named individuals working concurrently each on a series of individual tasks. Steps were conducted sequentially, with progress on the next step only beginning once all the tasks within the current step were completed. Steps were grouped into sections, with four levels Section, Step, Person, Task thus rigidly defined.

The fixed hierarchy was found wanting, however, when we attempted to articulate the detail within a step, or needed to aggregate procedures at larger scales.

The first problem arose when trying to describe the specific actions to be taken within a task. The idea of a fifth level of hierarchy, Detail, was considered as the proper name for a sub-task, but the rigid hierarchy was showing its cracks. If a procedure was only one person doing many detailed actions,

Keywords: Procedures, Complexity Measure, Fractal Analysis, Systems Engineering, Hyperbolic Space

A. F. Cowie, “Turtles All The Way Up: the Complexity and Hyperbolic Nature of Procedures” in [Preprint]. Sydney, NSW: Harrington Free Press, Dec. 2025.

the additional structure necessary to group it became onerous. Why did we even need a name for this level?

The second problem was one of composition. One of the original insights of the "Surviving Change" paper [3] and the subsequent "Massive Changes and Upgrades" seminars based on it [4] was the importance of summarizing a procedure when presenting to stakeholder audiences. While it was possible to extract a summary from the outline of a procedure, at the end of the day different levels of an organization will consider different magnitudes of action to be a "step". A senior manager might consider there are four things to do; an engineer tasked with the third one knows that to do that third step requires them to perform eight.

Our initial effort to build software to support procedures in this style [2] ran into both these problems and several other challenges. It was our first foray into the question of how to write such procedures in a versionable and repeatable fashion. The work raised the question of where procedures were to be stored, how they were to communicate results, how they should be composed, and how interactions with the external physical world could best be described.

Grappling with how to represent work at different scales led us to the realization that the structure of work at all such scales was the same.

Fundamental Unit of Procedure

Definition

A procedure represents the instructions to accomplish a task. Such a task is not random action, but directed work taking place to meet the intentions of the person, team, organization, or purpose that needs the outcome to make progress toward their goals [5].

To accomplish anything, the entities that perform the work in this procedure have to:

1. **Wake** up
   (get woken up or otherwise notice there's work to do)

2. **Depend** on
   0, 1, or many children

3. Then, optionally
   Do something

4. **Store** state

5. (as a convenience) **Reply** to parent

This unit of work is called a *step*.

Theorem *The Fundamental Unit of Procedure.* Any entity performing work in a procedure: activates (by being signalled or otherwise detecting it has work to do). It depends on zero, one, or more "children", waiting for all to complete. Then, optionally, it performs a task. The entity then durably stores the state of its being completed along with a result value from its task, if any. Finally, it signals back to its parent indicating it is complete.

This defines a procedure as a recursive container. Because children are themselves steps, steps depend on other steps. That structure has depth in the relationships between steps, and breadth, in the number of tasks a given procedure encompasses. As such a procedure is a tree structure.

The most atomic element within the definition of a step is the part where it durably stores its result and the fact of its completion. At its most basic, the act of executing a step is actually recording that it has taken place. In other words, **a step is something that can be observed as having been done**.

While the verb "record" is unfortunately suggestive of bureaucracy and paperwork, at its simplest the notion of recording is a synonym for the creation of information.

The use of the word durable here does not mean "carved in stone", but rather is used in the sense of persistence: what is created is a new piece of information that represents something *has occurred*. In other words, without the observation of this fact—and the ability to communicate that fact—the fact hasn't happened. Creating or representing a piece of new information is, by Shannon's Theory of Information, the transmission of (at least) one bit [6]. Everything else is context.

The final part's use of the word "signal" suggests active outwards communicating, but if the parent can read the durable states written by its children's steps and watch for those states to change, then the act of a child writing the outcome is equivalent to the child signalling the parent.

Composition

Steps can be composed in parallel or in sequence.

A number of steps that are to be carried out simultaneously can be written

$$\{a_1|a_2|...|a_n\}$$

which is one unit of structure for n tasks. Such tasks are unordered and can be executed independently and in parallel if the context so allows, but there is no dependency between any of these tasks and they can record their results in any order.

A procedure that depends on a single step (which in turn can depend on a single step, which in turn…) is collectively a set of steps that run in series. To show this ordering we write

$$\{c, \{b, \{a\}\}\}$$

As a notational convenience we can use

$$\{a, b, c\}$$

to represent dependencies that must be carried out sequentially, but the first expression shows that ordering requires an addition of structure equal to the number of tasks to be done in order.

Finally, a step with no dependencies is a leaf node in the procedure

$$\{a\}$$

denoted a *task*,

$$a$$

which brings us back to the definition of step.

Endeavours of any complexity can be articulated as a composition of steps and tasks. The space of all possible procedures is $\mathcal{P}$.

Procedures are Complex

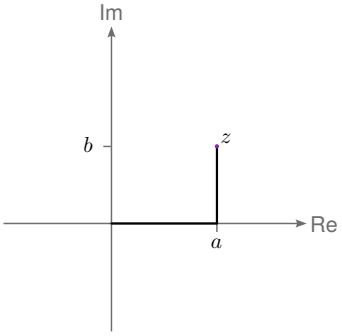
A simple checklist with dozens of tasks in a single step is quite different from a detailed procedure with phases composed of sections comprised of steps assigned to people each with many tasks, but both are valid procedures.

By the *Fundamental Unit of Procedure* any procedure $\mathbf{p}$ in $\mathcal{P}$ can be expressed as a tree structure. Each step contains zero, one, or more tasks; those steps with tasks are leaf nodes of a tree. Each step has zero, one, or more dependencies; steps with dependencies are nodes in the tree. We can assess the complexity of a procedure by describing it in terms of the number of tasks performed and the depth of the tree structure those tasks are composed into. These two quantities are orthogonal. This leads directly to describing a procedure's complexity as a single complex number.

A complex number $z \in \mathbb{C}$ is a quantity that has a real component, a, and an imaginary component, b, such that

$$z = a + ib$$

where $a, b \in \mathbb{R}$ and i is the square root of -1. The number can be represented as a point on the Cartesian plane taking a in the $\hat{\mathbf{x}}$-axis and b in the $\hat{\mathbf{y}}$-axis.



The number can also be expressed in polar coordinates

$$\begin{aligned} z &= r(\cos\theta + i\sin\theta) \\ &= re^{i\theta} \\ &= r\angle\theta \end{aligned}$$

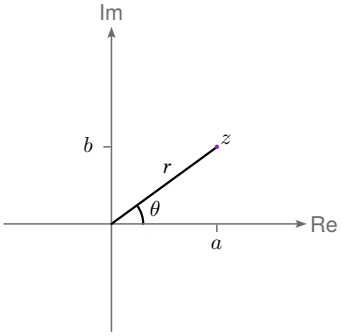
where r is the magnitude

$$\begin{aligned} r &= |z| \\ &= \sqrt{a^2 + b^2} \end{aligned}$$

and θ the argument

$$\begin{aligned} \theta &= \arg(z) \\ &= \tan^{-1}\left(\frac{b}{a}\right) \end{aligned}$$

which can likewise be represented graphically.



For any procedure we can count the number of actual tasks, and likewise we can express the structure of the procedure by counting the number of steps, in other words the number of structural nodes.

$$\begin{aligned} \mathit{tasks} &= \sum_i \text{tasks}^\star(\mathbf{p}_i) \\ \mathit{steps} &= \sum_i \text{deps}^\star(\mathbf{p}_i) + 1 \end{aligned}$$

where $\text{tasks}^\star$ is a recursive function that counts the number of tasks in a given node $\mathbf{p}_i$ and its dependencies

$$\text{tasks}^\star(\mathbf{p}_i) = \text{tasks}(\mathbf{p}_i) + \text{tasks}^\star(\text{deps}(\mathbf{p}_i))$$

with $\text{deps}^\star$ being similarly defined.

We are not just interested in the raw number of tasks, but in the amplitude of the amount of work and the structure used to organize and coordinate that work. Thus for a procedure we have the following definition:

$$\mathit{complexity} = \mathit{tasks} + i\,\mathit{steps}$$

which is a projection of $\mathbf{p}$ onto $\mathbb{C}$. Many different procedures can have the same shape and so would have the same complexity.

To establish the size of a procedure we begin by taking its complex magnitude

$$|\mathit{complexity}| = \sqrt{\mathit{tasks}^2 + \mathit{steps}^2}$$

then scale it to place magnitude 1 on the unit circle, thereby defining the *size* of a procedure:

$$p = |\mathit{complexity}| \times \sqrt{2}$$

A procedure of size 1 is thus a fundamental step with one task and no dependencies.

The complex argument of the complexity, taken as a phase angle, describes the balance of the procedure in terms of the relationship between work described and the structure of that work

$$\begin{aligned} \theta &= \arg(\mathit{complexity}) \\ &= \tan^{-1}\left(\frac{\mathit{steps}}{\mathit{tasks}}\right) \end{aligned}$$

Expressing θ in units of half π-radians we have the *shape* of the procedure:

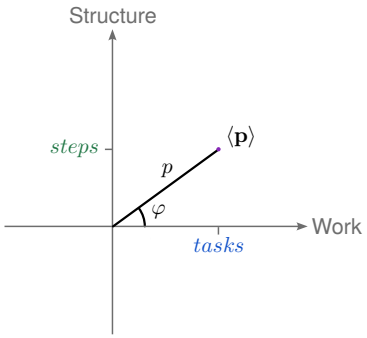
$$\varphi = \theta \times \frac{2}{\pi}$$

which scales the phase angle to be in a range between 0 and 1. An unordered collection of unrelated tasks would be shape 0 (but is not actually a procedure, corresponding to $\tan^{-1}\theta$ being undefined at $\frac{\pi}{2}$), whereas a procedure with all steps and no actual tasks would be shape 1 (such a procedure could be observed, but executing it would be effectively instantaneous so also not particularly useful).

Taken together, the complexity of a procedure can be written

$$\langle \mathbf{p} \rangle = p \angle \varphi$$

where p is the size of a procedure and φ is its shape.



Scale Independence

We propose the unit for the size of a procedure is the *turtle*, adopting symbol `p`.[1] The fundamental step with one task and no dependencies is one turtle in size, written `1 p`.

Unlike other fundamental and derived units in the Système Internationale [7], the turtle is *scale independent*. At any given time a procedure is composed of a single step whose structure is as described here. But any step can refer to sub-steps which can themselves be procedures. Thus the turtle is a *recursive unit*.

Ultimately though, the number of steps executed to carry out a procedure can be counted in base 10, and so a moderately involved procedure taking 100 steps to complete at all levels could be described as being a hectoturtle in size, written `1 hp`. The turtle is thus a measure of complexity; extremely large processes and operations composed of thousands of procedures with millions of steps would be described in megaturtles or gigaturtles.

The dependencies of a step are themselves steps. It can be difficult to know in advance the underlying internal complexity of a task, thus the utility of using submultiples of the turtle unit is questionable. By definition a nanoturtle represents one of the billions of actions that ultimately make up the execution of the procedure at hand, but enumerating all of its siblings might prove challenging in practice.

If, however, you can enumerate the fractional substeps that compose into a step, then there is no difference between a `100 p` procedure whose leaf steps are composed of microturtle-sized substeps,

$$\begin{aligned} &\log(100) - \log(10^{-6}) \\ &= 8 \end{aligned}$$

and a procedure of size `100 Mp`,

$$\begin{aligned} &\log(100 \times 10^{6}) \\ &= 8 \end{aligned}$$

The size of the procedure depends on the position of the observer, but the complexity is the same regardless. These examples allow us to define the *scale* of a procedure:

$$\langle\!\langle \mathbf{p} \rangle\!\rangle = \log\langle \mathbf{p} \rangle - \log\langle \mathbf{p}_n^{-1} \rangle$$

where $\mathbf{p}$ is the procedure as viewed from the perspective of the observer and $\mathbf{p}_n$ is the smallest of the submultiples that compose that procedure.

[1] We would have been delighted to recommend `t` to the *Comité Consultatif des Unités* as the symbol for the fundamental procedural unit in SI but unfortunately `t` was already taken for tonne, the convenience unit equivalent to `1 Mg`. So the final consonant of "step" (or equivalently the first of "procedure") was selected as the symbol for the turtle.

Fractal Nature of Procedures

In the Introduction we described the scenario where a project manager at one level considers that a task may require four steps to be completed, while the software developer assigned one of those tasks has the knowledge that performing that task requires a large number of complex actions. How do we assess the complexity? The manager thinks the process is 4 tasks in 1 step, `5.68 p`, whereas the engineer knows it is $4 + 8$ tasks in 2 steps, `16.97 p`.

Both answers are correct; **complexity depends on the observer's point of view**. The question resolves itself if we consider what impact knowledge of subordinate tasks has on the structure of the procedure. In any given branch of a procedure tree there is a current known smallest level of work, and tasks at this level have an amplitude of 1. All the structure of the procedure is built up from there, and so $complexity$ is always > 1.

If, however, there are steps to be undertaken that are outside of view beneath one of these tasks and composed into it, then those interior tasks have $0 < complexity < 1$. If a step thought to be a leaf node $\{a\}$ turns out to be comprised of n equal interior substeps, then each of those substeps contributes one nth of a task to the whole. Ten substeps would each contribute 0.1 turtles; if those substeps in turn have ten sub-substeps, then they would make up 0.01 of the parent task, and so on. A one turtle task could be composed of a million microturtles or a trillion picoturtles of work, but the enclosing task remains at `1 p` regardless.[2] The decomposition of a procedure thus occupies a fractal space and so the turtle is also a *fractal unit*.

If the manager at the higher level in our example considers that the procedure is only four tasks, then four tasks it is, and management remains blissfully unaware of the detail required to actually get the job done. The greater the detail, the further away it is from having an impact on the observer.

Equally, our aspiring captain of industry may not entirely realize their actions take place in a larger setting. As such their work is composed into larger structures they may not be aware of. In a very real way, the contribution their work makes to that larger whole fades into the larger gestalt and their ability to grasp the complexity of their environment fades away just as rapidly.

Thus a procedure is everything, but it takes place in contexts which are both larger and smaller, which are themselves procedural in nature. Procedures are fractal, but we can still measure the complexity of any given procedure, and compare them.

System Equivalence

A procedure is a composition of steps that allows us to define the instructions needed to accomplish an outcome and to observe it having been accomplished. A system, on the other hand, is a composition of elements which, given requirements and inputs, produces outputs [8]. The difference is the interaction of elements, the physicality of the process, and its cyclic ongoing nature [9]. A system exists.

The complexity of a system is a property of *systems design*; what a system does is its own best definition, but our understanding and description of the intent of the system is something we can analyze. The complexity of a system of interest in domain D has its complexity assessed as having an objective component and a subjective component [10], where the subjective component is from the point of view of an observer.

$$K_D(C) = \left\{ K_D^O(C), K_{D,A}^S(C) \right\}$$

[2] Note that the symbol for *turtle*, p, is not to be confused with the *pico* prefix, symbol p.

A system does not have to be in the most optimal configuration to be valid; if it performs to its requirements then it is sufficient. The Efatmaneshnik Elegance of a system [11] compares the complexity of the problem to all known sufficient designs j:

$$E_j = \frac{C_p}{C_{s_j}}$$

Of the sufficient designs for a systems one is the least complex, and is thus the most elegant.

We can apply the same consideration to procedures when viewed as systems: many articulations of a procedure will accomplish a given output. The one with the lowest complexity is the most elegant, which again brings us to consideration of the size of a procedure, established above. The elegance of a procedure is

$$E_j = \frac{\langle \mathbf{p} \rangle}{\langle \mathbf{p}_{s_j} \rangle}$$

for all $\mathbf{p}_{s_j} \in \mathcal{P}_s$, the set of procedures that are sufficient to achieve the requirements S.

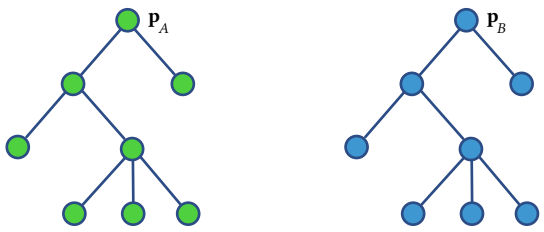
This essay into elegance show us a limitation: there is no point in comparing a four-point executive summary with a thousand-step detailed plan because they are clearly different magnitudes. Procedures can only be compared if they have similar scales. In other words

$$\mid \langle\!\langle \mathbf{p} \rangle\!\rangle - \langle\!\langle \mathbf{p}_{s_j} \rangle\!\rangle \mid < \varepsilon$$

with comparisons only being meaningful where ε is less than 1.

Procedures Form a Hyperbolic Space

We established above that procedures are tree-like structures. All procedures in $\mathcal{P}$ are unique, but two distinct procedures $\mathbf{p}_A$ and $\mathbf{p}_B$ can have the same structure [12].



That is, if two procedures recursively have the same number of steps, each with a matching number of substeps and tasks,

$$\text{tasks}^\star(\mathbf{p}_A) = \text{tasks}^\star(\mathbf{p}_B)$$
$$\text{deps}^\star(\mathbf{p}_A) = \text{deps}^\star(\mathbf{p}_B)$$

then

$$\langle \mathbf{p}_A \rangle = \langle \mathbf{p}_B \rangle$$

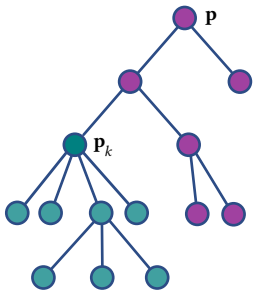
Equivalently, we can use the approach of calculating the edit distance [13] to compare the difference between two trees, such that if

$$\text{dist}(\mathbf{p}_A, \mathbf{p}_B) = 0$$

then we say these two procedures are *congruent*, written

$$\mathbf{p}_A \cong \mathbf{p}_B$$

We want to know where a procedure $\mathbf{p}$ is located in $\mathcal{P}$, and where a dependency $\mathbf{p}_k$ is in relation to it.



Each procedure $\mathbf{p}$ in $\mathcal{P}$ can be uniquely identified by its compositional path from some origin procedure $\mathbf{p}_0$.

We define this as a procedure with no tasks and no structure that can always be observed to have been completed, so

$$\psi(\mathbf{p}_0) = 0$$

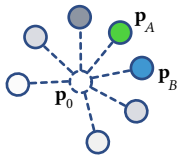
and

$$\langle \mathbf{p}_0 \rangle = 0$$

making it the identity procedure under composition

$$\mathbf{p} \circ \mathbf{p}_0 = \mathbf{p}$$

If two procedures are congruent, then they are equidistant from the origin.



This works for two independent procedures, but procedures contain other procedures. Procedures that differ in scale should be farther apart than procedures that differ only in size at the same scale. A procedure that contains $\mathbf{p}$ should be further away. From the perspective of the enclosing parent, those children are "smaller". From the perspective of a child procedure, it may not even be aware of the "larger" structure beyond it.

We describe the position of a procedure as

$$\psi : \mathcal{P} \to \mathbb{D}$$

If procedure $\mathbf{p}$ has a dependency $\mathbf{p}_k$, then the position of that procedure in $\mathcal{P}$ is some small distance δ_k further away from the origin

$$\psi(\mathbf{p}) = \psi(\mathbf{p}_k) + \delta_k$$

but since the increase in compositional distance from a child to a parent is a single step, δ_k is always 1. We can generalize; the distance from the origin to a procedure is proportional to the complexity added by all of the composition steps needed to get there. Therefore,

$$\psi(\mathbf{p}) = \tanh(|\langle \mathbf{p} \rangle|)e^{i\theta}$$

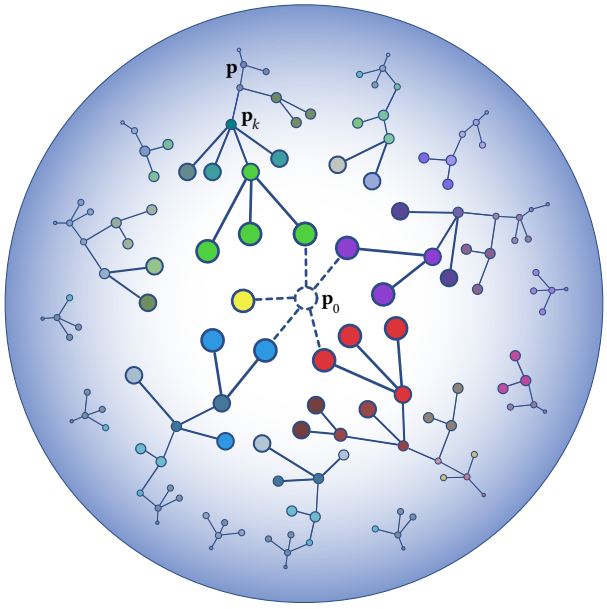
where θ here distinguishes among the many procedures at the same compositional depth. The magnitude places all procedures of depth n at the same distance from $\mathbf{p}_0$, whereas the argument spreads them around that circle. As n increases, the number of possible procedures grows exponentially: the larger the procedure, the more ways it can be composed from an ever-increasing number of component parts.

The set of procedures at compositional distance n from the origin grows exponentially with depth:

$$|\mathcal{P}_n| \sim c^n$$

for some $c > 1$. This exponential growth with distance is the defining characteristic of a hyperbolic geometry, where volume grows exponentially. Finally, when we compose procedures (one procedure calling another as a sub-procedure), the relationship is multiplicative in complexity but additive in log-scale distance.

Tree structures are intrinsically hyperbolic in the sense of Gromov [14]. The compositional structure of $\mathcal{P}$ is a tree, with path distance as the natural metric. Having renormalized, all procedures have a distance from the origin that is less than 1. $\mathbb{D}$ is therefore the open unit disk, and we can consider this projection of $\mathbf{p}$ as the Poincaré disk model with the boundary representing procedures of unbounded compositional depth—forever approachable but unreachable.



Thus $\mathcal{P}$ is a hyperbolic space.

Conclusion

Our original notion of a rigid structure for a procedure was found wanting when we attempted to articulate the detail within a step, or needed to aggregate procedures at larger scales than the hierarchy supported. Examining these two extremes uncovered the fact that a procedure is scale independent, leading to the definition of a fundamental unit of procedure presented here.

It is usually considered that infinite depth is “turtles all the way down”, but we have seen here that the turtle is a unit from which larger procedures can be built. No matter how humble a task, it can be composed with other such tasks in ever larger combinations to ultimately achieve a grand endeavour; each step along the way is itself a procedure that is part of a larger whole. It really is turtles all the way up.

Acknowledgements

The outline of these ideas was first described in [2] and [4] and were subsequently developed in discussions with Peter Miller, Lindsay Holmwood, Silvia Pfeiffer, Erik de Castro Lopo, and Robert Collins culminating in the original presentation of the *Fundamental Unit of Procedure* at [15]. This version has been updated with the analysis of the complexity of procedures, consideration of their fractal nature, and further valuable feedback from Edward Kmett, J. R. Lynn, and Adam McCullough. The insights and encouragement from all these colleagues have long been greatly appreciated.

References

[1] T. Pratchett, *The Light Fantastic*. Colin Smythe, 1986.

[2] A. F. Cowie, “Realizing a Graphical User Interface for Procedures,” in *CodeCon 2007*, Peter Miller and Company, Sep. 2007. doi: 10.31224/5768.

[3] A. F. Cowie, “Surviving Change,” in *Systems Administrator Guild of Australia Annual Conference Proceedings*, SAGE-AU, Aug. 2004. doi: 10.31224/5596.

[4] A. F. Cowie, “Mastering Massive Changes and Upgrades to Mission-Critical Systems,” in *Proceedings of the 19th Conference on Systems Administration (LISA '05)*, D. N. Blank-Edelman, Ed., USENIX, Dec. 2005.

[5] J. A. Dextraze, “The Art of Leadership,” in *Canadian Forces Personnel Newsletter*, Department of National Defence, Jun. 1973.

[6] C. E. Shannon, “A Mathematical Theory of Communication,” in *The Bell System Technical Journal*, 1948, pp. 379–423, 623–656.

[7] BIPM, *The International System of Units (SI)*. Bureau International des Poids et Mesures, 2019. [Online]. Available: https://www.bipm.org/documents/20126/41483022/SI-Brochure-9-EN.pdf

[8] D. Walden, G. Roedler, K. Forsberg, R. D. Hamelin, and T. Shortell, Eds., *INCOSE Systems Engineering Handbook*. John Wiley & Sons, 2015.

[9] C. L. Magee and O. L. de Weck, “Complex System Classification,” in *Proceedings of the 14th Annual International Symposium of the International Council on Systems Engineering*, INCOSE, 2004, pp. 533–546. doi: 10.1002/j.2334-5837.2004.tb00510.x.

[10] M. Efatmaneshnik and M. J. Ryan, “A General Framework for Measuring System Complexity,” *Complexity*, vol. 21, no. S1, pp. 533–546, 2016, doi: 10.1002/cplx.21767.

[11] M. Efatmaneshnik and M. J. Ryan, “On the Definitions of Sufficiency and Elegance in Systems Design,” *IEEE Systems Journal*, vol. 13, no. 3, pp. 2077–2088, 2019, doi: 10.1109/jsyst.2018.2875152.

[12] M. H. Hamilton, “What the Errors Tell Us,” *IEEE Software*, vol. 35, no. 5, pp. 32–37, 2018, doi: 10.1109/MS.2018.290110447.

[13] T. Akutsu, T. Mori, N. Nakamura, S. Kozawa, Y. Ueno, and T. N. Sato, “On the Complexity of Tree Edit Distance with Variables,” in *33rd International Symposium on Algorithms and Computation (ISAAC 2022), Leibniz International Proceedings in Informatics (LIPIcs)*, Schloss Dagstuhl — Leibniz-Zentrum für Informatik, 2022, pp. 44:1–44:14. doi: 10.4230/LIPIcs.ISAAC.2022.44.

[14] M. Gromov, “Hyperbolic groups,” in *Essays in Group Theory*, S.M. Gersten, Ed., Mathematical Sciences Research Institute Publications, 1987, pp. 75–263. doi: 10.1007/978-1-4613-9586-7_3.

[15] A. F. Cowie, “Towards a Fundamental Structure for Tasks,” in *CodeCon 2011*, Peter Miller and Company, Oct. 2011. doi: 10.31224/5838.