For the sake of simplicity: Applying software design parsimony to the content of information system ontologies

Timothy Tambassi
Ca' Foscari University of Venice

Abstract

Although many information system ontologies [ISOs] claim to be parsimonious, the notion of parsimony seems to influence the debate on ISOs only at the level of vague and uncritical assumption. To challenge this trend, the paper aims to clarify what it means for ISOs to be parsimonious. Specifically, section 2 shows that parsimony in computer science generally concerns software design and, together with elegance, is one of the two aspects of the broader notion of simplicity. Section 3 transforms the main claims of parsimony in software design into claims about the content of ISOs, the combination of which is hereafter called "parsimony of content"—where "content" refers only to the content of ISOs. Sects. 4-7 discuss the application of this parsimony to the design of ISOs, and outline different kinds (and combinations) of parsimony of content. Finally, section 8 considers whether parsimony of content could provide some criteria both for selecting and/or classifying the contents of ISOs and for choosing between different and equally consistent ISOs.

Keywords

information system ontologies, ontological aims, parsimony, representation primitives, simplicity.

Philosophical Problems in Science (Zagadnienia Filozoficzne w Nauce)
No 75 (2023), pp. 135–155

> *There are two ways of constructing a software design: one way is to make it so simple that there are obviously no deficiencies, and the other way is to make it so complicated that there are no obvious deficiencies.*
> Tony Hoare (1980)

1. Introduction

According to Turner (2018), there are two methodological advantages to adopting parsimony in software design:

- diminishing the amount of work,
- reducing the risk of error.

«This is in line with Quine, who, in the case of theories, argues that parsimony carries with it pragmatic advantages, and that pragmatic considerations themselves provide rational grounds for discriminating between competing theories» (Turner, 2018, p.139).

Acknowledging such advantages, however, does not imply that the adoption of parsimony is mandatory. Indeed, in speaking of information system ontologies [ISOs], Smith (2004) and Grenon (2008) remark that nothing prevents ISOs from:

[1] endorsing/rejecting different assumptions,
[2] including parsimony among those assumptions,
[3] considering the possibility of multiple forms of parsimony, and then repeating [1–2].

Despite [1–3], the adoption of parsimony is so common for ISOs that many ISOs implicitly and uncritically assume this notion. To prevent parsimony from influencing the debate on ISOs at the level of

an implicit and uncritical assumption, this paper aims to clarify what it means for ISOs to be parsimonious. Sect. 2 shows that parsimony in computer science generally concerns software design and, together with elegance, is one of the two aspects of the broader notion of simplicity. Sect. 3 transforms the main claims of parsimony in software design into two claims about the contents of ISOs, the combination of which is hereafter called "parsimony of content"—where "contents" refers only to the contents of ISOs. Sects. 4–7 discuss the application of this parsimony to the design of ISOs, and outline different kinds (and combinations) of parsimony of content. Finally, Sect. 8 considers whether parsimony of content could provide some criteria both for selecting and/or classifying the contents of ISOs and for choosing between different and equally consistent ISOs.

2. Parsimony in software design

One of the main reasons why computer scientists place simplicity at the core of good and/or successful software design[1] is that simplicity contributes to the transparency and reliability of the design.[2] According to Turner (2018, pp.133–134), simplicity does not have a single meaning in this context; rather, it refers to two distinct and related notions: elegance (or syntactic simplicity) and parsimony (or ontological simplicity).[3]

---

[1] On software design, see Allen (1997); Baljon (2002); Parsons (2015).

[2] On simplicity in software design, see also Wirth (1974); Dijkstra (1979).

[3] See also Baker (2016), who analyzes the distinction between elegance and parsimony within the philosophy of science debate.

Elegance generally concerns the graspability, clarity, transparency, correctness, efficiency, consistency, generality, uniformity, and explanatory power of software.[4] Parsimony links software design with its specification[5], and insists that

[4] software solutions do not go beyond *what is required*.

While Turner further specifies the meaning of "what is required" in [4] by claiming that

[5] software should solve the problem it aims to solve, but no more,

Pawson (1998) takes one step further. First, he considers

[6] parsimony to have been achieved when it is no longer possible to improve software by subtraction.

Then, he adds that

[7] parsimony is the quality that software applications have when their components, details, and junctions have been reduced to the essential.

[7] in turn means that

[8] the link between the design and the aims of software (see [4–5]) also concerns the components, details, and junctions of the software.

[4] On elegance in software design, see Bentley and McIroy (1993); Gelernter (1998); Oram and Wilson (2007); Hill (2018); Turner (2018).

[5] One referee rightly pointed out that there are other ways of relating simplicity and parsimony. The example they give is simplicity in understanding the code (i.e. "semantic simplicity"), including self-commenting code, which is simple in terms of understanding the code. I fully agree with them. I can only note here that this paper is not intended to exhaust the debate on the relationship between simplicity and parsimony. For more details on semantic simplicity, see Gelernter (1998); Sober (2002); Turner (2018).

[4–8] (together) imply that

[9] parsimony concerns the [9.1] aims of software and [9.2] its components, details, and junctions.

3. Parsimony in information system ontologies

Section 2 has shown that:

[10] simplicity is at the core of good and/or successful software design;
[11] simplicity can be divided into elegance and parsimony.

Turner (2018, p.128) adds that

[12] design is everywhere in computer science.

This means that, if [10–12] hold, parsimony also applies to the design of ISOs.

Gruber (2009) defines ISOs as follows:

[13] ISOs are sets of representational primitives (henceforth, primitives) with which to model a domain (of knowledge).[6] Primitives are primarily instances, classes, properties, and relations.[7]

[6] For further (and competing) definitions of ISO, see Neches et al. (1991); Gruber (1993); Guarino and Giaretta (1995); Bernaras et al. (1996); Borst (1997); Swartout et al. (1997); Studer et al. (1998); Guarino (1998); Uschold and Jasper (1999); Sowa (2005); Noy and McGuinness (2003); Tambassi and Magro (2015). Gruber (2009, p.1964) has also affirmed that ISOs, or "ontologies", are artefacts specified by (ontological) languages. Before him, Guarino and Giaretta (1995) have pointed out that "ontology" in computer science has (at least) two different meanings: the artefact and the philosophical discipline—which finds direct application in computer science (see, for example Turner, 2018; Krzanowski and Polak, 2022). This explains why "ontology" can have the same meaning in both philosophy and computer science.

Therefore, based on [4–9], applying parsimony (of software design) to [13] means that:

[14] ISOs should not go beyond the problem(s) they aim to solve (see [4–5] and [9.1])—that is, beyond the domain(s) (of knowledge) ISOs aim to model;
[15] the components, details and junctions of ISOs, that is the primitives of ISOs, should be reduced to the essential (see [6–7] and [9.2]).

Henceforth, by "parsimony of content" (where "content" refers only to the contents of ISOs) I will mean the application of [4–9] to [13], namely [14–15]. There are two main reasons for this emphasis on "content"—rather than on "parsimony of ISOs" in the broader sense. The first reason is that, within the debate on ISOs, the notion of parsimony is chiefly associated with the content of primitives.[8] Therefore, to speak of "content" in "parsimony of content" and "primitives" in [13] (i.e. according to Gruber's definition of ISO) means to account for this relation. The second reason is that parsimony of content does not (aspire to) exhaust the debate on the parsimony of ISOs. In other words, there may *in principle* be other parsimonies involved in the ISOs debate, as well as other ways of applying [4–9] to ISOs. And this is also in line with [10–12], which do not rule out that parsimony

[7] Instances are the lowest-level components, the basic units, of ISOs (Laurini, 2017). Classes, which may contain sub-classes and/or be sub-classes of other classes, are sets of instances that share common features (Jaziri and Gargouri, 2010). Properties describe the various features of a class and of its instances (Noy and McGuinness, 2003; Jaziri and Gargouri, 2010). Relations represent the way in which both classes and instances interact with each other (Laurini, 2017). On primitives, see also Tambassi (Tambassi, 2021).

[8] See Burgun et al. (1999); Yao et al. (2011); Motara and Van der Schiff (2019); Partridge et al. (2020).

could be "everywhere" in ISOs, and thus also apply to something other than the content of ISOs (Turner, 2018, pp.161–167). Moreover, although it would transitively follow from [8–9] that

[16] parsimony of content deals with *both* [14–15],

we should also consider the possibility of

[17] following [14–15] separately.

Indeed, if adopting parsimony of content means following both [14–15] (see [16]), nothing prevents us from adopting parsimony of content partially, that is, from adopting either [14] or [15] by itself.

4. On the rivers of the UK

To specify what it means to adopt parsimony of content in practice, suppose we build an ISO, ISO_1, aimed at [A_1] listing and [A_2] classifying all the rivers of the UK. Unless A_1 and A_2 are further specified, A_1 is fulfilled if and only if

[18] no river in the UK is excluded from ISO_1,

whereas achieving A_2 means

[19] providing any classification of such rivers.

[18] generally refers to the notion of completeness (of ISOs),[9] according to which

[9] See Bittner and Smith (2008).

[20] the contents of an ISO should be exhaustive[10] with respect to the domain that the ISO aims to model.

For ISO_1, [20] means that the nearly 1,500 rivers crossing the UK should find their place among the contents of ISO_1, which ultimately fall within (one of) the primitives of ISO_1 (see also [13]), no matter which primitive.

As for [19], A_2 can in principle be achieved in many ways. For example, ISO_1 could

[21] classify the rivers according to their biotic and/or topographic features;
[22] systematize the rivers according to the geographical region(s) they cross;
[23] catalogue the rivers according to some (arbitrary) length intervals;
[24] consider [21–23] together;
[25] provide any arbitrary classification.

The reason why there can be many ways to achieve A_2 is that A_2 does not specify any criteria for classifying the UK's rivers. Therefore, to the extent that each of [21–25] classifies the UK's rivers, there is no way to prefer one among [21–25] over the others, at least according to A_2.

[10] See Tambassi (2021b). "Exhaustive" in [20] also refers to the debate on categories in philosophy, within which "exhaustive" represents one of the three criteria of adequacy (see Cumpa 2019), indicating that whatever there is (or could be) should find its place in one and only one category (see Thomasson 2019).

5. On the aims of information system ontologies

According to [14–15], applying parsimony of content to ISO_1 entails that:

[26] ISO_1 should not go beyond its aims;
[27] (and) the primitives of ISO_1 should be reduced to the essential.

As for [26], ISO_1 has two aims: A_1 and A_2. In accordance with [26], ISO_1 is thus expected to

[28] list all the UK's rivers (see A_1),
[29] classify those rivers (see A_2),
[30] do nothing more than what [25–26] specify.

[28] implies [18], [29] leads to [21–25], and thus assumes that there can be different ways of fulfilling [26], or that ISO_1 could not go beyond its aims in different ways. [30] limits ISO_1's tasks to [28] (or A_1) and to [29] (or A_2). This means that, according to [30], ISO_1 should not, for example,

[31] list the UK's lakes,
[32] (or) classify Germany's rivers,

because [31–32] would go beyond A_1 and A_2, and hence contradict [26] and [28–29]. All this also implies that

[33] (all) ISO_1's contents should be consistent with and functional to its aims,

but also that

[34] no content of ISO_1 should go beyond the aims of ISO_1.

However, things can get complicated in cases like the following. Suppose we fulfill [29] by means of [23], that is, by classifying the UK's rivers according to some (arbitrary) length intervals, such as 0–40 miles, 40–80 miles, 80–120 miles, and so on. What about the property "length of the river"? Does the inclusion of such a property within ISO_1's contents follow from [33–34]? On the one hand, one could answer "no": A_1 and A_2 only require [28–29], which do not explicitly refer to the specific length of the rivers. On the other hand, one could also answer "yes", insofar as the "length of the river" would justify the assignment of each (UK) river to one of the length intervals of [23].

6. Completeness and parsimony of content

The principle of completeness (of ISOs) states that the contents of an ISO should be exhaustive for the domain that the ISO aims to model (see [20]). Applying completeness to (ISO_1's) A_1 implies [14], but does not exclude that:

[35] the same river appears twice (or several times) in ISO_1,
[36] ISO_1 also includes the UK's lakes and/or Germany's rivers.

Conversely, applying parsimony of content to A_1 implies [18], but excludes [36] because of [33–34]—which are ultimately inferred from [26]. From [36], however, it does not follow that completeness and parsimony of content are mutually contradictory, since:

[37] ISOs may consistently follow both completeness and parsimony of content (see [1]).

To justify [37], we could consider the negation of [36] to be only a possibility for completeness, as well as a necessity for parsimony of content. The same, we may add, can be said for the negation of [35]. If so,

[38] how does the negation of [35] follow from parsimony of content?

To answer [38], let us return to [27], according to which ISO_1's primitives should be reduced to the essential. If [27], an (easy) solution might be to avoid repetitions, so that

[39] each content of an ISO should appear only once in the same ISO.

Now, [39] is based on [27], which follows from [15], which in turn is one of the two pillars of parsimony of content (see [14–15]). Moreover, maintaining [39] means affirming the negation of [35], which is a necessity for parsimony of content and a possibility for completeness. But if so, [37] can also be justified by [35].

7. Parsimony of content and (representational) primitives

While [39] follows from [27], this is not all. Indeed, "ISO_1's primitives should be reduced to the essential" seems to be open to different interpretations, such as:

[40] reducing the types of the primitives we use (to the essential);

[41] reducing the tokens of the primitives we use (to the essential);[11]
[42] combining [40] and [41].

To explain [40–42], let us imagine that ISO_1 follows [23] and thus classifies all the UK's rivers according to some (arbitrary) length intervals. $ISO_1 \wedge$ *[23]* therefore has two aims: (A_3) to list all the UK's rivers and (A_4) to classify them according to [23].

Now, [40] suggests reducing the types of primitives: using fewer primitive types to model a domain (see [13] and [20]) is preferable to modelling the same domain using more primitive types. This means that placing [S_1] the UK's rivers among the instances of $ISO_1 \wedge$ *[23]* and the intervals of length among the classes of $ISO_1 \wedge$ *[23]* (respectively) would be preferable to [S_2] placing those rivers and length intervals among the instances, classes, and properties of $ISO_1 \wedge$ *[23]*. Indeed, S_1 uses fewer primitive types than S_2.

[41] is instead ambiguous. It may refer to

[41.1] an ISO's overall amount of tokens,

meaning that the tokens of $ISO_1 \wedge$ *[23]* should be reduced to the essential. Now, while A_3 (simply) requires that all of the nearly 1,500 rivers crossing the UK find their place among the contents of $ISO_1 \wedge$ *[23]* (for example, among the instances of $ISO_1 \wedge$ *[23]*), A_4 might be fulfilled in different ways. Supposing, for example, that each length interval corresponds to a class of $ISO_1 \wedge$ *[23]*, [41.1] suggests that

[11] The distinction between [40] and [41] is largely based on Fiddaman and Rodriguez-Pereyra's (2018) distinction between two different forms of ontological economy. According to the authors, the principle of qualitative economy requires us to avoid multiplying types of entities when not necessary, while that of quantitative economy requires us to avoid multiplying token entities when not necessary. For further reading on ontological economy, see Sober (1975), Lewis (1973), van Inwagen (2001), Lando (2010), and Schaffer (2015).

[S_3] classifying the UK's rivers by means of two length intervals (e.g., "longer than 100 miles" and "shorter than 100 miles") is preferable to [S_4] classifying those rivers by means of five length intervals (e.g., "between 0–30 miles", "between 40–80 miles", "between 80–120 miles", and so forth). Why so? Because S_3 requires (almost) 1,500 instances and 2 classes, 1,502 tokens in total, whereas S_4 requires (almost) 1,500 instances and 5 classes, 1,505 tokens in total. (This also means that, insofar as S_3 and S_4 are both consistent with the aims of $ISO_1 \wedge [23]$, it is irrelevant to [41.1] whether S_4 is more detailed than S_3). But [41.1] also represents a way of balancing the overall tokens of $ISO_1 \wedge [23]$ within the various primitives. For example, if achieving A_3 and A_4 required that S_3 also include 10 properties and S_4 includes 2 properties, then S_4 would be preferable to S_3. In other words, the reduction of tokens within one primitive should not be at the expense of a proliferation of tokens within the whole ISO.

[41.1], however, is not the only way to interpret [41], which might also refer to

[41.2] the tokens of each primitive.

In turn, [41.2] could have two interpretations: [41.2.1] and [41.2.2]. [41.2.1] indicates that modelling $ISO_1 \wedge [23]$ by means of, for example, [S_5] 10 classes, 2 relations and 1,500 instances is preferable to modelling $ISO_1 \wedge [23]$ by means of [S_6] 2 classes, 3 relations, 2 properties and 1,500 instances. For although there is no difference between S_5 and S_6 in terms of the tokens of instances, *and* S_6 is preferable to S_5 in terms of the tokens of classes, S_5 is preferable to S_6 in terms of the tokens of relations and properties. This means that, according to [41.2.1], we should prefer S_5 over S_6, insofar as S_5 is preferable with regard to both relations and properties, and S_6 is preferable only with regard to classes. [41.2.2] instead focuses on

the tokens of each primitive, the reduction of which is independent from one primitive to another, and does not directly concern [41.1] or [41.2.1]. In other words, [41.2.2] offers the chance to apply [41] (and more generally [15] or [27]) to one and only one primitive. Consequently, we could have a [41] based on tokens of classes when the tokens of classes are reduced to the essential, a [41] based on the tokens of relations when the tokens of relations are reduced to the essential, and so on for each primitive. All this does not imply that those applications of [41] to one and only one primitive cannot be combined to improve [15], [27] and/or [40–41], nor that the list of primitives will never change, and with it the varieties of applications of [41] to which the primitives refer.

However, there are also ambiguities surrounding [42]. Firstly, it is unclear whether

[43] the combination refers to [40] and [41.1], or [40] and [41.2.1], or [40] and [41.2.2], or [40], [41.1], and [41.2.1], and so on.

Secondly,

[44] once [43] is clarified, we should also define the order of priority of the combination.

To clarify [44], let us suppose that the combination refers to [40] and [41.1]. Giving priority to [40] means that reducing the types of primitives is more important than reducing the total number of tokens: that is, both primitive types and tokens should be reduced to the essential but the reduction of the tokens comes after that of the types of primitives. Giving priority to [41.1] means the opposite.

8. Parsimony (of content) as a set of criteria

According to [14], ISOs should not go beyond their aims, whatever these may be. As regards the contents of an ISO, [14] means that they should all be consistent with the ISO's aims (see [33–34]). According to [15], for any ISO, we should reduce the types of primitives (see [40]), the total number of tokens (see [41.1]), or the tokens of each primitive (see [41.2.1] and [41.2.2]) to the essential. Alternatively (see [42]), we could adopt [40] and one or more of [41.1], [41.2.1], and [41.2.2] by defining their priority. According to [16], we should adopt both [14] and [15], or better [14] and at least one of [40], [41.1], [41.2.1], [41.2.2], or [42].

On this basis, let us focus on ISO_1's A_2, according to which ISO_1 should provide a classification of the UK's rivers. Now, insofar as A_2 does not specify any criteria to classify the UK's rivers and [21–25] are (all) consistent with A_2, there is no reason why we should not regard

[45] [21–25] as *equally consistent* with A_2.

But, if [45], how are we to choose among [21–25]? The fact that the criteria, if any, are not deducible from A_2 does not imply or guarantee that [14–16] provide any criteria. In other words,

[46] choosing among [21–25] may both [46.1] (at least partially) depend *and* [46.2] not depend on (some of) [14–16].

In turn, [46] does not imply or guarantee that

[47] once we choose among [21–25], [14–16] provide criteria for selecting and/or classifying the contents of ISOs.

All this means that parsimony of content (in general) can provide:

[48] some criteria for choosing among different and equally consistent classifications/ISOs;

[49] some criteria for selecting and/or classifying the content of ISOs;

[50] both [48] and [49];

[51] neither [48] nor [49].

9. Concluding remarks

Since some ISOs adopt parsimony as an implicit and uncritical assumption, and/or without explaining what parsimony specifically consists of (or refers to), these pages sought to clarify the point. In this regard, I introduced the notion of parsimony of content, showing that

[52] this parsimony concerns two main claims, [14–15], as well as their connection, [16], from which [33–34], [37], [39–40], [41.1], [41.2.1], [41.2.2], [43–44] and [48–51] follow.

[52] broadly suggests that the adoption of parsimony of content has to do with

[53] the interpretation and combination of claims about parsimony of content,

[54] specifying whether parsimony of content provides some criteria for choosing among different classifications/ISOs and/or for selecting and/or classifying the contents of ISOs.[12]

[12] Unlike some computer scientists (Floyd, 1967), I have not considered the possibility of combining parsimony of content with modularization: indeed, breaking down complex ISOs into (in)dependent modules would simply defer the question of adopting parsimony of content to both complex ISOs and their (in)dependent modules.

All this means that

[55] the notion (and application) of parsimony of content is multifaceted;
[56] an informed adoption of parsimony of content requires [53–54].

It does not follow from [55–56] that parsimony of content exhausts the debate on the parsimony of ISOs, nor that ISOs are bound to adopt parsimony of content. In other words, [55–56] are consistent with [1–3], thus ensuring the plurality of the methodological approaches shaping the debate on ISOs.

Funding. This paper has received funding from the European Research Council under the European Union Horizon Europe Research and Innovation Programme (GA no. 101041596 ERC—PolyphonicPhilosophy).

Disclamer. Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

Bibliography

Allen, R.J., 1997. *A Formal Approach to Software Architecture (CMU Technical Report CMU-CS-97-144)*. (technical report). Pittsburgh: Carnegie Mellon, School of Computer Science.

Baker, A., 2016. Simplicity. In: E.N. Zalta, ed. *The Stanford Encyclopedia of Philosophy*. Winter 2016. Metaphysics Research Lab, Stanford University. Available at: <https://plato.stanford.edu/archives/win2016/entries/simplicity/> [visited on 29 January 2024].

Baljon, C.J., 2002. History of history and canons of design. *Design Studies*. Philosophy of design, 23(3), pp.333–343. https://doi.org/10.1016/S0142-694X(01)00042-4.

Bentley, J.L. and McIlroy, M.D., 1993. Engineering a sort function. *Software: Practice and Experience*, 23(11), pp.1249–1265. https://doi.org/10.1002/spe.4380231105.

Bernaras, A., Laresgoiti, I. and Corera, J., 1996. Building and Reusing Ontologies for Electrical Network Applications. In: W. Wahlster, ed. *Proceedings of the 12th European Conference on Artificial Intelligence (ECAI'96)*. Chichester, UK: John Wiley and Sons, pp.298–302.

Borst, W.N., 1997. *Construction of Engineering Ontologies for Knowledge Sharing and Reuse*. PhD thesis. University of Twente, Centre for Telematics and Information Technology (CTIT). Available at: <https://research.utwente.nl/en/publications/construction-of-engineering-ontologies-for-knowledge-sharing-and-> [visited on 31 January 2024].

Burgun, A., Botti, G., Fieschi, M. and Le Beux, P., 1999. Sharing knowledge in medicine: semantic and ontologic facets of medical concepts. *IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No.99CH37028)*. Vol. 6. Tokyo, Japan: IEEE, pp.300–305. https://doi.org/10.1109/ICSMC.1999.816568.

Dijkstra, E.W., 1979. The Humble Programmer. Turing Award Lecture. In: E. Yourdon, ed. *Classics in Software Engineering*. New York, N.Y: Yourdon Press, pp.113–128. Available at: <http://archive.org/details/classicsinsoftwa00your> [visited on 31 January 2024].

Fiddaman, M. and Rodriguez-Pereyra, G., 2018. The razor and the laser. *Analytic Philosophy*, 59(3), pp.341–358. https://doi.org/10.1111/phib.12128.

Floyd, R.W., 1967. Assigning meanings to programs. *Proceedings of Symposium on Applied Mathematics*, 19, pp.19–32. Available at: <http://laser.cs.umass.edu/courses/cs521-621.Spr06/papers/Floyd.pdf>.

Gelernter, D.H., 1998. *Machine Beauty: Elegance and the Heart of Technology*. New York: Basic Books. Available at: <http://archive.org/details/machinebeautyele00gele> [visited on 31 January 2024].

Grenon, P., 2008. A Primer on Knowledge Representation and Ontological Engineering. In: K. Munn and B. Smith, eds. *Applied Ontology*. Frankfurt am Main: Ontos Verlag, pp.57–82. https://doi.org/10.1515/9783110324860.57.

Gruber, T.R., 1993. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2), pp.199–220. https://doi.org/10.1006/knac.1993.1008.

Gruber, T., 2009. Ontology. In: L. Liu and M.T. Özsu, eds. *Encyclopedia of Database Systems*. Boston, MA: Springer US, pp.1963–1965. https://doi.org/10.1007/978-0-387-39940-9_1318.

Guarino, N., 1998. Formal Ontologies and Information Systems. In: N. Guarino, ed. *Formal Ontology in Information Systems. Proceedings of FOIS'98, Trento, Italy, 6-8 June 1998.* Amsterdam: IOS Press, 3–15.

Guarino, N. and Giaretta, P., 1995. Ontologies and Knowledge Bases: Towards a Terminological Clarification. In: N.J.I. Mars, ed. *Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing*. Amsterdam: IOS Press, pp.25–32.

Hill, R.K., 2018. Elegance in Software. In: L. De Mol and G. Primiero, eds. *Reflections on Programming Systems*. Vol. 133, *Philosophical Studies Series*. Cham: Springer International Publishing, pp.273–286. https://doi.org/10.1007/978-3-319-97226-8_10.

Jaziri, W. and Gargouri, F., 2010. Ontology Theory, Management and Design: An Overview and Future Directions. In: W. Jaziri and F. Gargouri, eds. *Ontology Theory, Management and Design: Advanced Tools and Models*. IGI Global, pp.27–77. https://doi.org/10.4018/978-1-61520-859-3.ch002.

Krzanowski, R. and Polak, P., 2022. The meta-ontology of AI systems with human-level intelligence. *Philosophical Problems in Science (Zagadnienia Filozoficzne w Nauce)*, (73), pp.199–232.

Lando, G., 2010. *Ontologia. Un'introduzione*. Rome: Carocci.

Laurini, R., 2017. *Geographic Knowledge Infrastructure: Applications to Territorial Intelligence and Smart Cities*. London: ISTE Press - Elsevier.

Lewis, D., 1973. *Counterfactuals*. 1st ed. Oxford: Blakwell.

Motara, Y.M. and Van Der Schyff, K., 2019. A functional ontology for information systems. *South African Computer Journal*, 31(2). https://doi.org/10.18489/sacj.v31i2.691.

Neches, R. et al., 1991. Enabling technology for knowledge sharing. *AI Magazine*, 12(3), pp.36–36. https://doi.org/10.1609/aimag.v12i3.902.

Noy, N.F. and McGuinness, D.L., 2003. *Ontology Development 101: A Guide to Creating Your First Ontology*. (technical report). Stanford, CA: Stanford University. Available at: <http://www.ksl.stanford.edu/KSL_Abstracts/KSL-01-05.html> [visited on 1 February 2024].

Oram, A. and Wilson, G., eds., 2007. *Beautiful code*. 1st ed, *Theory in practice series*. Beijing; Sebastapol, Calif: O'Reilly.

Parsons, G., 2015. *The Philosophy of Design*. Cambridge: Polity press.

Partridge, C. et al., 2020. *A Survey of Top-Level Ontologies - to inform the ontological choices for a Foundation Data Model*. (technical report). CDBB. https://doi.org/10.17863/CAM.58311.

Pawson, J., 1998. *Minimum*. London: Phaidon Press.

Schaffer, J., 2015. What not to multiply without necessity. *Australasian Journal of Philosophy*, 93(4), pp.644–664. https://doi.org/10.1080/00048402.2014.992447.

Smith, B., 2004. Ontology. In: L. Floridi, ed. *The Blackwell Guide to the Philosophy of Computing and Information*. 1st ed, *Blackwell philosophy guides*, 14. Malden, Mass.: Blackwell, pp.155–166.

Sober, E., 1975. *Simplicity*. Oxford: Oxford University Press. https://doi.org/10.1093/acprof:oso/9780198244073.001.0001.

Sober, E., 2002. What is the Problem of Simplicity? In: A. Zellner, H.A. Keuzenkamp and M. McAleer, eds. *Simplicity, Inference, and Modelling*. Cambridge: Cambridge University Press, pp.13–32.

Sowa, J.F., 2005. *Guided Tour of Ontology*. Available at: <http://www.jfsowa.com/ontology/guided.htm> [visited on 29 January 2024].

Studer, R., Benjamins, V.R. and Fensel, D., 1998. Knowledge engineering: Principles and methods. *Data & Knowledge Engineering*, 25(1), pp.161–197. https://doi.org/10.1016/S0169-023X(97)00056-6.

Swartout, B., Patil, R., Knight, K. and Russ, T., 1997. Toward Distributed Use of Large-Scale Ontologies. *AAAI Symposium on Ontological Engineering*. Stanford, CA, pp.138–148.

Tambassi, T., 2021. *The Philosophy of Geo-Ontologies: Applied Ontology of Geography*, *SpringerBriefs in Geography*. Cham: Springer International Publishing. https://doi.org/10.1007/978-3-030-78145-3.

Tambassi, T. and Magro, D., 2015. Ontologie informatiche della geografia. Una sistematizzazione del dibattito contemporaneo. *Rivista di Estetica*, (58), pp.191–205. https://doi.org/10.4000/estetica.447.

Turner, R., 2018. *Computational Artifacts: Towards a Philosophy of Computer Science*. Berlin, Heidelberg: Springer. https://doi.org/10.1007/978-3-662-55565-1.

Uschold, M., Box, P.O. and Usa, W., 1999. A Framework for Understanding and Classifying Ontology Applications. *Proceedings of the IJCAI99 Workshop on Ontologies and Problem-Solving Method*. Stockholm.

Van Inwagen, P., 2001. *Ontology, identity, and modality: essays in metaphysics*, *Cambridge studies in Philosophy*. Cambridge, U.K.; New York: Cambridge University Press.

Wirth, N., 1974. On the Design of Programming Languages. *Proc. IFICP Congress 74*, pp.386–393. Available at: <https://web.eecs.umich.edu/~bchandra/courses/papers/Wirth_Design.pdf> [visited on 1 February 2024].

Yao, L. et al., 2011. Benchmarking ontologies: Bigger or better? *PLoS Computational Biology*, 7(1), e1001055. https://doi.org/10.1371/journal.pcbi.1001055.