Musing On Datums

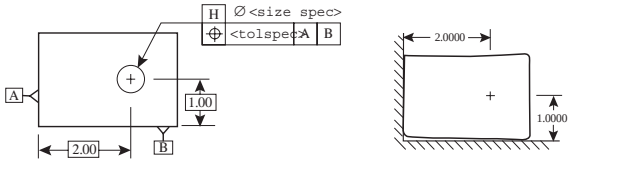
by Herb Voelcker

Datum referencing in design and CMM practice is mainly simple, but special cases, and rules for handling them, continue to proliferate and complicate the practice and teaching of GD&T. Here we explore a different way of thinking about datums, and a specification system that has no rules.

Let me begin with a caution. This article uses symbology and concepts from American and International Standards in non-standard ways. Readers should draw no inferences on Standard Practice from this article, and publication of the article implies no recognition or review by any standards organization.

Now let's get to work. Datums are used to construct datum reference frames (DRFs), and DRFs are used to specify the relational properties of part features – typically position and orientation. The 2-D example in Figure 1 illustrates the main ideas, which most readers will know well. Specifically, the true position of hole H in Fig. 1a is established on the actual part in Fig. 1b by dimensions, at "gagemakers' precision," anchored to simulators of the datum features A and B. The simulators could be the faces of a 90° angle fixture; the part is "immobilized" in the fixture by establishing stable (2-point, in 2-D) contact with the A-face, and then 1-point contact with the B-face…to reflect the datum ordering in the position (⌖) tolerance callout in Fig. 1a.

Let's put a question on the table before leaving Fig. 1. Clearly the A and B simulators are linear because the A and B datum features are linear, but why are the simulators ⊥ (perpendicular)? Because A and B are ⊥ in the part design? Perhaps…but the orthodox answer is: because the Standards say so.



(a) Partial specification

(b) Establishing the true position of hole H.

Figure 1: A simple example.

Figure 2 provides an example of a 'tertiary datum problem'…a generic problem that has vexed standards committees for a decade. The task here is to establish the true position of hole D, using the Basic dimension and the ordered datums A (planar), B (cylindrical), and C (slot), with B and C at material condition RFS, i.e. maximal fitted size.[1]

The simulators for the B and C datums are each ⊥, by rule, to the simulator for the primary A-datum, and so we can represent the simulator geometry by the 2-D drawings in Figure 3. In Fig. 3a, the axis of the B simulator lies in the midplane of the C simulator, whereas in Fig. 3b the axis and midplane are independent (but ⊥A); in both geometries, the B and C simulators are at maximal size. Observe also that the 3b case admits the two dimensioning patterns shown in Figures 3b-1 and 3b-2. Which of the three cases – 3a, 3b-1, or 3b-2 – is correct? I leave that as an exercise for the reader: ferret it out of the standards, while noting that a designer might want to specify any one of the three cases.

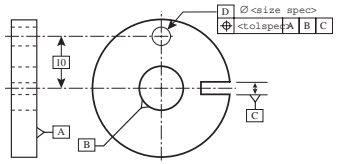


Figure 2: An example posing a "tertiary datum problem."

DRF declarations that admit several plausible interpretations are not difficult to construct. Generally these cases are resolved to a single 'right answer' by increasingly complex rules. The underlying problem is that the classical 'callout block' symbolism for defining DRFs is not powerful enough to distinguish multiple interpretations. The clean way to fix the problem is to redesign the declaration language under a strong backward compatibility constraint, but doing only that, without some attendant conceptual changes, seems to be difficult for both technical and political reasons.

*

In the remainder of this article we'll explore a different approach to DRF declaration and construction. Be warned that we'll start and end with clean, simple concepts, but there will be some rather tedious clutter in the path from start to finish.

Let's begin by abandoning almost all of the current rules governing datums and DRFs; we'll keep only the notions of datum ordering (precedence) and material condition (MMC, etc.) for features of size. We'll also abandon strategies based on 'immobilizing' parts by degree-of-freedom reduction, and replace them with a strategy based on 'fitting' datum simulators to datum features one-by-one, with each fit (except the first) subject to constraints that must be stated explicitly…because we have discarded all interpretative rules.

Here's how it works in the example of Figure 1. The datum callout A|B in Fig. 1 is replaced with A|B⊥A, which means, "fit a linear simulator to the A feature in the usual manner, and then fit a linear simulator to B that is also ⊥ the A-simulator." Figure 4 illustrates the steps. (Fig. 4c is included merely for contrast: in our new world of explicit constraints, it corresponds to A|B.) The DRF we want is developed in Fig. 4b and is the same as that shown in Fig. 1b, but now we know why the simulators are ⊥: we specified ⊥. Why we did so should be evident later.

For meatier examples, we'll write DRF declarations for the geometries in Fig. 3: see the box at the bottom of this page. We'll start with the Fig. 3b declaration in the box; it is simpler than the 3a declaration, because B and C are independent in Fig. 3b illustration.

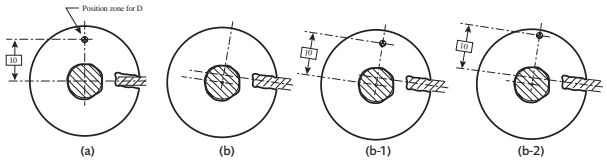


Figure 3: Datum simulator geometry for Fig. 2.

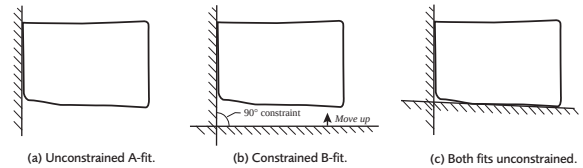


Figure 4: Establishing a DRF for the part in Fig. 1a.

The 3b declaration is read as follows:

- A : induce the primary A-simulator without external constraints.
- B : induce the B-simulator at RFS ("B @ S ") *and* (symbol "∧") B ⊥ A .
- C : similar to B.

This declaration (for figure 3a) is similar to the one above, but now a third constraint has been added for the C-simulator: that its centerplane contain (symbol "⊃") the axis of the B-simulator.

Fig. 3b-1 & 3b-2: The dimensional ambiguity shown in these drawings is a red herring (something that draws attention away from the main issue)! Clearly there is ambiguity – one cannot determine from Fig. 2 whether the Basic dimension is anchored on the B or the C feature – and in current practice it is resolved by appealing to a DRF rule. However, the problem is *not* a datum problem: it arises from ambiguity in a drawing, and would not arise in a modern CAD system, where dimensions are associated with surface features rather than projections. The cleanest way to solve the problem in drawings is to tag Basic dimensions: see Figure 5.

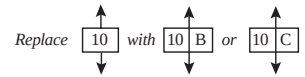


Figure 5: Resolving referential ambiguity in Figure 2.

*

What have we accomplished thus far? We have a new paradigm for establishing datum simulators (sequential fitting under constraints), we have a math-like language for writing constraints, and we're rule-free. It's a powerful and flexible combination…*but it won't be used unless we can find a way to cut through all that constraint clutter.* We need some kind of 'constraint shorthand.'

Here is a solution. *All of the constraints we're likely to need* – relations like ⊥ and ⊃ , distance in the form of Basic dimensions, and a few more – *are embedded in the design geometry of the part*; all we need is a procedure that enables us to use the constraints selectively …and here is such a procedure.

DRF DECLARATIONS FOR FIGURES 3A AND 3B

Figure 3b: A | (B@ S) ∧ (B⊥A) | (C@ S) ∧ (C⊥A)

Figure 3a: A | (B@ S) ∧ (B⊥A) | (C@ S) ∧ (C⊥A) ∧ (Centerplane(C) ⊃ Axis(B))

[1] RFS is shorthand for Regardless of Feature Size, which is defined (for present purposes) as the largest perfect cylinder that will fit into a hole, or the largest perfect slab that will fit into a slot. All of the examples in this article use Feature of Size datums at RFS – often denoted explicitly by Ⓢ, for clarity – because that is usually the most stringent condition.

DRF Induction Procedure

- Induce (establish, fit) the primary datum simulator without external constraints (as at present).
- Induce the secondary simulator under all constraints between the primary and secondary datums carried in the design (nominal, ideal) geometry of the part.
- Induce the tertiary simulator similarly, i.e. under all constraints between the primary, secondary, and tertiary datums carried in the design geometry of the part.
- The foregoing steps may be overridden by explicit constraint statements in the datum block. A simple constraint language must be provided for overrides.

*

That's it. The procedure rests on the following simple premise. Every nominal feature in a proper part design is fully constrained; the constraints are carried in the part geometry, and thus we don't need to write them out explicitly. Datum simulators induced as above are as fully constrained as possible with respect to already established datums…unless the designer overrides the design-geometry constraints. Concisely, one simply 'does what the geometry says' or, in the case of overrides, what the designer says. The 'simple constraint language' turns out to be very easy when constraints are to be cancelled, which is almost always the situation. A simple cross-out syntax is used in the following examples.

To see how this works in practice, let's rewrite the DRF callouts for the Fig. 3 geometries. The first example will be explained in detail, to insure that readers understand the logic. Later examples can be solved by inspection, because the concepts we are using are very simple…once you grasp them.

Fig. 3a & 6a: | A | B S | C S |

The induction logic is shown in Table 1.[2] Simulator B is governed by all relations in the part between A and B, which are just ⊥ and a material condition for B. The C-simulator is governed by all relations in the part geometry between A, B, and C: these are shown in the last row of Table 1. The simulator geometry is shown in Fig. 3a and Fig. 6a, which duplicates 3a.

SIMULATOR	ORDER OF INDUCTION	EXTANT SIMULATORS	CONSTRAINTS (from part geometry and material condition specs)
A	1	none	none
B	2	A	(B ⊥ A) ∧ (B @ RFS)

Table 1: Induction logic for | A | B S | C S |

Fig. 6b: | A | C S | B S |

This declaration is a variant of that just above. The induction logic is very similar to that in Table 1 (loosely, exchange B and C in the last two rows). The resulting geometry is shown in Fig. 6b.

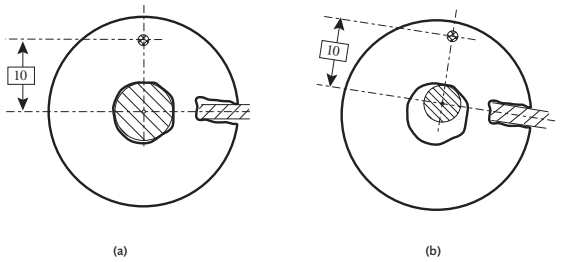


Figure 6: Fully constrained simulator inductions.

Fig 3b: | A | B S | C S | ~~B~~ | *or* | A | C S | B S | ~~C~~ |

Either of these declarations will produce the Fig. 3b geometry. Each has a cancellation – indicated by a slashed datum name – in the third slot. A cancellation means that constraints based on the cancelled datum are not to be used in inducing the current simulator (the tertiary simulator, in these cases)…and that is all a cancellation means. The effects of a cancellation are localized to a single simulator induction.

The induction logic for the first declaration is that in Table 1 with the B-dependency in the last row (the "∧ (C ⊃ B)" clause) removed, and similarly for the second declaration.

*

[2] The constraint C ⊃ B in the bottom right cell of Table 1 is shorthand for Centerplane (C) ⊃ Axis (B).

Figure 7 provides a final example. We focus again on the DRF for locating hole D.

Case 1: | A | B S | C |

This looks like a classic Y14.5 declaration. Fig. 8a shows induction of the B-simulator at RFS…no surprises here. Fig. 8b shows that the C-simulator cannot always be induced under all of the design geometry constraints (the Basic dimension imposes a distance constraint): this is a departure from traditional practice.

Case 2: | A | B S | C ~~B~~ | *or* | A | C | B S | ~~C~~ |

Either of these declarations produce the geometry in Fig. 9a, which corresponds to current practice.

Case 3: | A | C | B S | *…see Fig. 9b.*

***

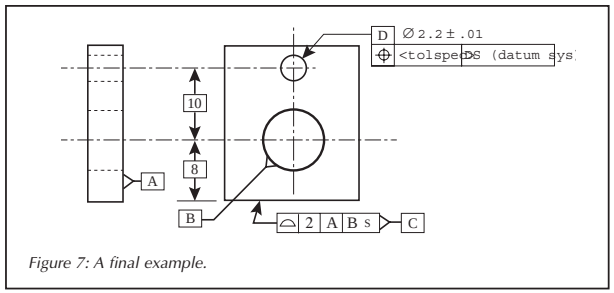


Figure 7: A final example.

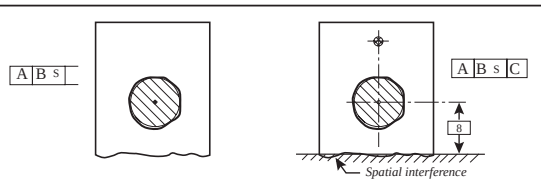


Figure 8a: B ⊥ A ∧ B @ RFS. *Figure 8b: C ⊥ A ∧ dist(Axis(B), C)) = 8.*

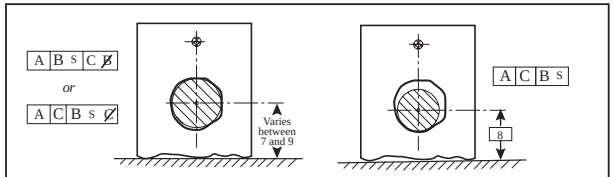


Figure 9a: Case 2. *Figure 9b: Case 3.*

There are more interesting topics to discuss – for example, constraint imposition (we have shown only cancellation) and coordinate-system induction. But this article is already too long, so I'll simply refer interested readers to the source document for this article, available at www.mae.cornell.edu/hbv/hbv01-5.pdf.

I am grateful to Alan Jones, Ed Morse, Al Neumann, Brian Shier, Walt Stites, and Bill Tandler for many past discussions that have influenced my views in this area. Ed Morse served as pre-publication reviewer of the article. ●

Herbert B. Voelcker is Graduate School Professor of Mechanical Engineering, and Charles Lake Professor Emeritus, at Cornell University. He can be reached at hbv1@cornell.edu.