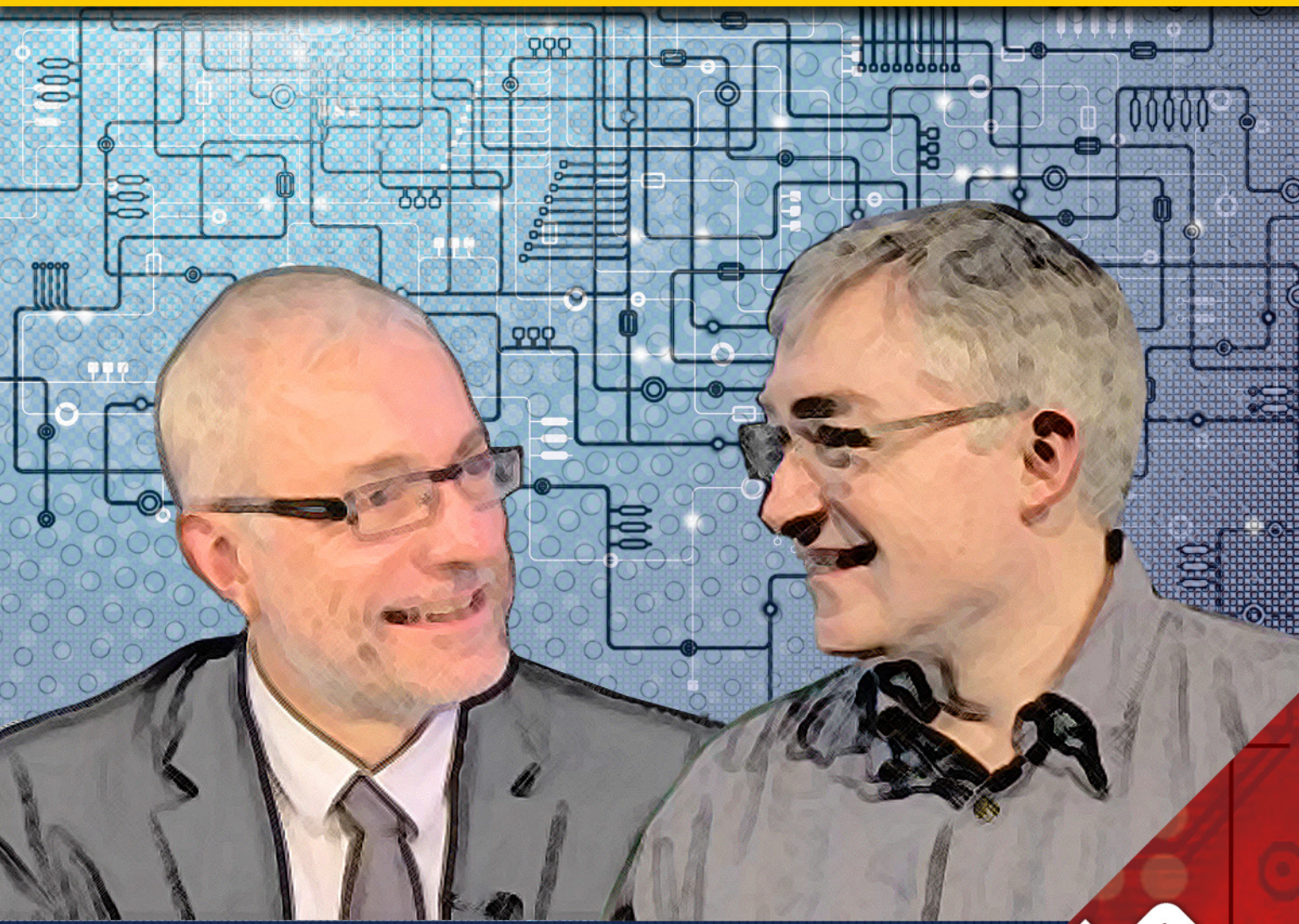
THE pcb design MAGAZINE

September 2014

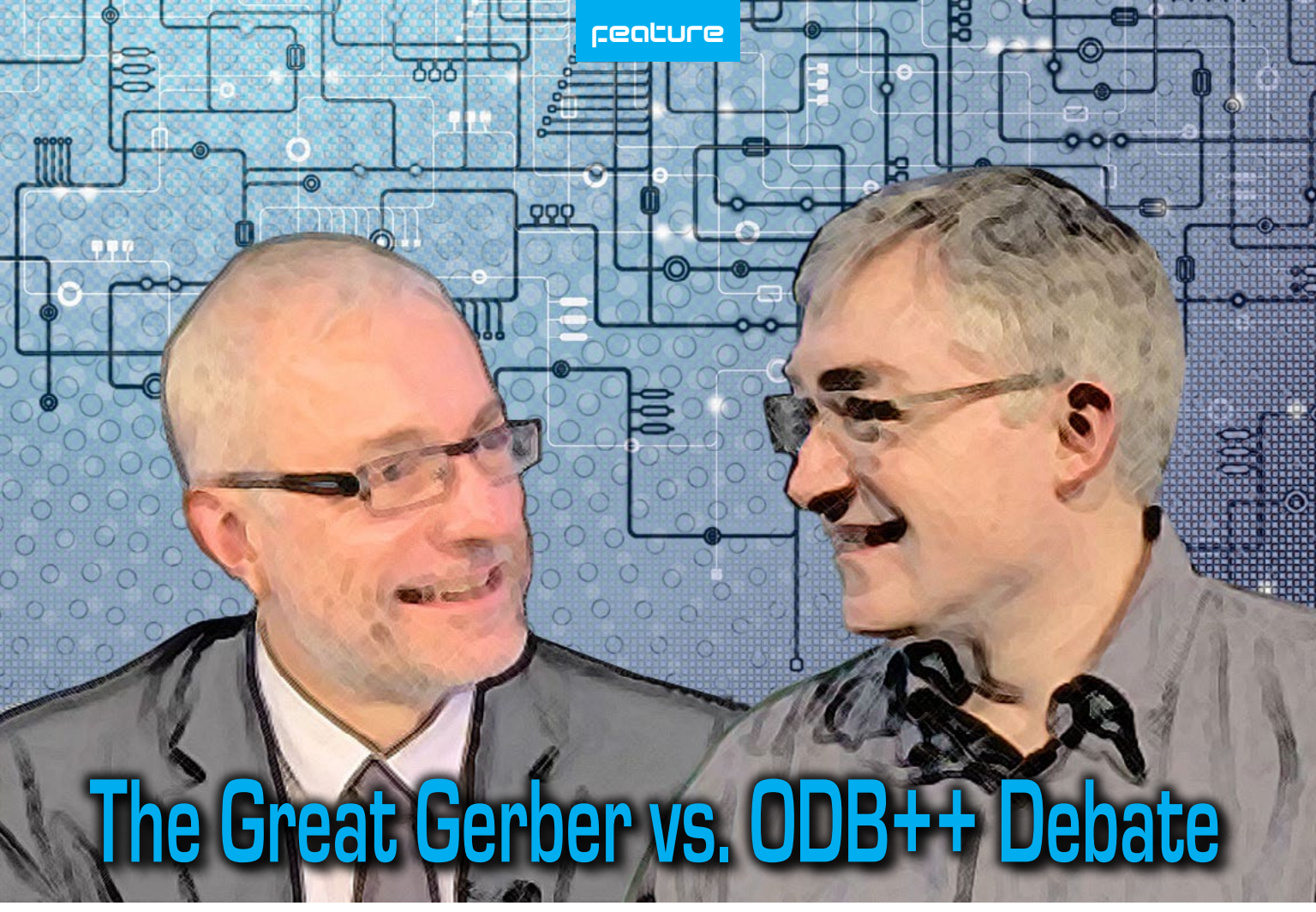
AN I-CONNECT007 PUBLICATION

DATA FORMATS



The Great Gerber vs. ODB++ Debate

page 12



The Great Gerber vs. ODB++ Debate

Editor's Note: This friendly debate (mostly friendly!) began with an article by Mentor Graphics' Julian Coates, which ran in the February issue of The PCB Magazine. *Karel Tavernier, managing director of Ucamco, which owns the Gerber format, replied to that article, and Coates was given the courtesy of a rebuttal so that they could be published side-by-side in the same issue. Finally, Tavernier replied to Coates, getting the last word...for now, at least.*

Gerber—the Smartest Way Forward

by Karel Tavernier

In a February 2014 article by Julian Coates of Mentor Graphics, Smart Data Formats Automate CAD/CAM, in which Coates promotes more widespread adoption of the ODB++ format, the arguments he uses indeed make it seem like OBD++ is the great panacea for our industry, one that promises to eliminate all problems for CAD-to-CAM data transfer without any downsides.

In order to promote ODB++, Coates unfortunately reverts to Gerber-bashing rather than explaining the strengths of ODB++. And his arguments are highly misleading, as they are based on some tired old fallacies that I would like to address here. Before starting, though, it's important to clarify that when referring to Gerber, I mean RS-274X Extended Gerber, the current Gerber format. This supersedes the earlier RS-274-D Standard Gerber format, which is obsolete. Bashing RS-274-D Standard Gerber is like railing against Windows because MS-DOS only allowed eight-character file names. If Coates wants to bash RS-274-D, I'll gladly join him. Having said this, less than 2% of all jobs are transferred using the old format, so it's practically a non-issue.

Extended Gerber is the PCB industry's de facto image data transmission format. New formats have come and gone; some, like the ODB++ format, have been around for decades, but still today, more than 90% of the world's PCBs, from

the simplest to the most complex, are still manufactured using Gerber, which tells me that this is an image format that the industry trusts. And the industry is right to trust it—it's the best there is. Used properly, it delivers on its promises, without fail, every time.

So let's have a look at some of Coates' arguments. He quotes Viasystems as stating that, "about 25% of the data packages they receive have issues relating to:

- Missing layers, fabrication drawings, drill files, etc…
- Netlist format violations
- Netlist exception violations."

If true, this is indeed a sorry state of affairs, and needs rectifying. But if this is the extent of the problems, then there is nothing wrong with the format. Viasystems' issues are in fact due to some rather trivial bugs in the CAD vendors' implementations, so the solution is to fix the implementations rather than to adopt completely new software by switching to a new format. The article is not clear about whether these omissions and violations relate to ODB++ or Gerber files, or a mix of the two. However they arise, I can only recommend that Viasystems report these issues to their customers with a request to contact their CAD software suppliers. If the CAD software vendors fix these simple bugs, the issues will be resolved once and for all. If they are unable or unwilling to do so, there is no solution: neither in Gerber, nor in ODB++, nor anywhere else for that matter.

Coates also mentions that Gerber files sometimes contain syntax errors, low numerical accuracy and other errors. This is no doubt true, but again these are simply bugs in the Gerber output. Do we need a new format to fix syntax errors in the current one? Surely the solution is to fix the bugs in the Gerber output. And ODB++ itself is not immune to syntax errors; if anyone would like some invalid ODB++ files, I can provide a few.

The reality is that Gerber files very rarely generate the wrong image. This is because while only a few applications read ODB++ reliably, there are countless more that read Gerber with near-perfect reliability. This is because:

- The Gerber format is simple
- Its specification is well-written, easy to read, detailed and precise
- Most of its implementations are mature
- As it is so widely used, the implementations are thoroughly field tested, so most bugs have been ironed out
- The format is supported by excellent free viewers such as GC-Prevue

Advocating the adoption of a new and much more complex format to eliminate simple bugs is a very curious solution indeed. Consider only that a CAD software developer struggling to produce a simple Gerber file correctly is not miraculously going to write a bug-free implementation for the more challenging ODB++ format. If one wants bug-free software it is best to stick with Gerber, as Gerber is a simpler and more mature format than ODB++, it is far less prone to bugs, and its bugs are far easier to find and resolve. Switching to a new imaging format introduces a whole raft of new issues and bugs that would take many years to sort out. Imaging software is complex and takes a long time to get right. Adopting ODB++ to solve bugs in Gerber output is like using a sledgehammer to swat a fly: The solution is far more damaging than the issue ever will be.

Table 1 summarizes Coates' claims regarding the benefits of ODB++ vs. Gerber.

Here is my take on the aforementioned benefits:

1. False. A simpler, more reliable format in fact needs less diagnostics
2. False. An error can be more easily identified in a simpler format.
3. False. ODB++ is not miraculously error-free.
4. False. IPC-356 supports the actual customer net name. It may be that the software Coates uses does not display it, but this then is a problem in that software.
5. False. Gerber has negative apertures and so can handle planes perfectly. (I should add that this is the first time I see the claim that ODB++ is more compact than Gerber!)

If these are the benefits of ODB++ and the reasons for adopting it, then Coates' argument collapses.

More importantly, Coates omits to mention the difficulties in adopting ODB++. Over the 20 years that ODB++ has been available, it has taken just 10% of the market share, with Gerber accounting for the remaining 90%. If ODB++ offers all the advantages espoused by Coates in his article, there can only be one of two reasons for its minimal uptake:

- The PCB industry consists largely of morons
- There are downsides to using ODB++

As I do not think this great industry is in the hands of morons, I believe that there must be some serious downsides to the adoption of ODB++. This is not because ODB++ is a particularly bad format: It is not. The point is that the adoption of ODB++ includes the adoption of a new image format, and image formats are notoriously hard to implement. Much has been written about just how complicated geometric software is and how much effort it takes to get it right, not to mention the years it takes to debug. So the implication that taking on this new image format is simple and low risk is at the very least misleading. Precisely because our industry's practitioners are not morons, they know this, so are reluctant to adopt a new format. They know very well how complex ODB++ is, and that it will give rise to many more problems, for many years.

The reality is that Gerber works very well for transferring images. In fact, there's nothing better.

Gerber X2

The most interesting point made by Coates is that Gerber files contain "no information about how the PCB layers stack up." This was a valid objection in the past, but it is no longer true, as the latest revision, Gerber X2, now contains layer stackup information.

At the heart of X2 is the use of attributes. These are akin to labels which provide information that are associated with image files, or features within them. The beauty of using attributes is that they are already familiar to CAM professionals and software developers, and they

sit naturally with the current capabilities of CAD and CAM systems. X2 extends the current Gerber specifiation with a series of standard attributes that are most important for efficient CAD-to-CAM communications, such as the function of each layer, whether a pad is a via or an SMD pad, and which are the component drill holes. As rather grandly stated elsewhere, X2 adds intelligence to the Gerber format. Software supporting X2 will read the whole Gerber archive automatically, with all layers in place, while identifying the function of each object.

Easy to adopt and to implement, X2 is upwardly compatible with the previous Gerber version. Altium, global leader in Smart System Design Automation, has been quick to recognise the value of X2 and will support it in an upcoming version of Altium Designer. By Q4 2014, Graphicode's widely-used and highly-respected GC-Prevue viewer will also support X2.

X2 maintains the trademark simplicity for which Gerber has always been known and used, and gives designers and engineers a standardized procedure that will require very little to change in their working practices—certainly none that would require approval, testing and all the rest. Equally important, this new revision does not disrupt existing workflows. If the software does not support the new capabilities, the old workflow continues to operate. Nobody is forced to buy anything. So this will be a very gentle, low cost improvement indeed, but the effects will be nothing short of revolutionary.

Coates omitted to mention this latest development in the Gerber format, one of the most

important developments in CAD-to-CAM automation today, given that it concerns the industry's de-facto standard format. Neither did he mention the alternative IPC-2581. Had he done so, his arguments for ODB++ might have been less compelling of course, but these omissions in an article titled Smart Data Formats Automate CAD/CAM lead to serious doubts about its objectivity.

Coates also added a diagram to the article comparing Gerber to ODB++ input in CAM. This compared a badly implemented Gerber with a well implemented ODB++. I have taken the liberty of adding a proper X2 Gerber to the schematic. The result, given in Figure 1, shows that if ODB++ is a smart format, Gerber X2 is a very smart one.

Conclusion

When CAD-to-CAM data sets use properly implemented Gerber archives, plus correct IPC-356-A files, problems in data transfer are rare. Where a problem or bug appears, the easiest, fastest and most economical solution is to fix it. This is because issues are not down to the format itself, but more likely due to its implementation in CAD software, and they are simple to resolve. The very worst solution would be to replace Gerber with the far more complex ODB++ format, because implementing a new format is never simple, quick, and/or risk free, especially when the new format is as complex as ODB++. The problems that would arise from such a move would be significant, and would hound the industry for many years.

The simplest, most practical path forward is to fix bugs in current implementations, and adopt Gerber X2 functionality.

One of the best things about this path is that it is incredibly kind on the industry, while enabling the PCB industry to benefit from all the advantages that ODB++ claims to deliver in Coates' article, but with none of the downsides. This is because it does not involve the wholesale adoption of a new format. Furthermore, the revised Gerber format is compatible with the previous versions of Gerber and older software, so improvements can be as gradual as users want them to be, with no one being forced to buy new software against their business wishes or budgetary constraints. It is a path that delivers to small software vendors and the industry at large, fixing what is broken without compromising what already works. In short, it's nothing short of revolutionary, but without the complications.

The Gerber format specification, a sample X2 archive and background articles on X2 can be found at www.ucamco.com/downloads.

Karel Tavernier
Managing Director, Ucamco

Julian Coates' Rebuttal:

With respect to Karel, I think he may be missing the main point. Consider this:

- No doubt Gerber is a very fine format for defining the graphical layers of a PCB
- IPC-D-356 is perfectly fine for defining a netlist

ODB++ Benefits for the PCB Fabricator
1. Import and export diagnostics are significantly reduced compared to Gerber
2. Errors can be identified and communicated to the customer much earlier in the process
3. Eliminates format errors and net exceptions that are common with Gerber
4. Fabricator can be allowed to see actual net name used by customer, easing the process
5. Less data is required for handling positive planes

Table 1.

- Excellon needs no improvement; it defines the location and diameter of drilled holes quite well.
- Component placement lists can define component positions and rotations quite well also
- PDF is a good format for rendering drawings
- GenCAD and FATF are good for defining the parts of a PCB assembly for testing purposes
- Word is good for capturing text, especially "Readme" documents that explain to a CAM engineer how all of the above file-types should relate to each other, and how to reintegrate all that data back together so as to enable an efficient software-driven new product introduction (NPI) process.

Certainly, if all you want is accurate graphical data, then I am sure Gerber meets the requirement, and Karel is to be congratulated on his perseverence in improving that particular 50-year-old NC format. At a recent industry debate on this topic, he suggested that the best

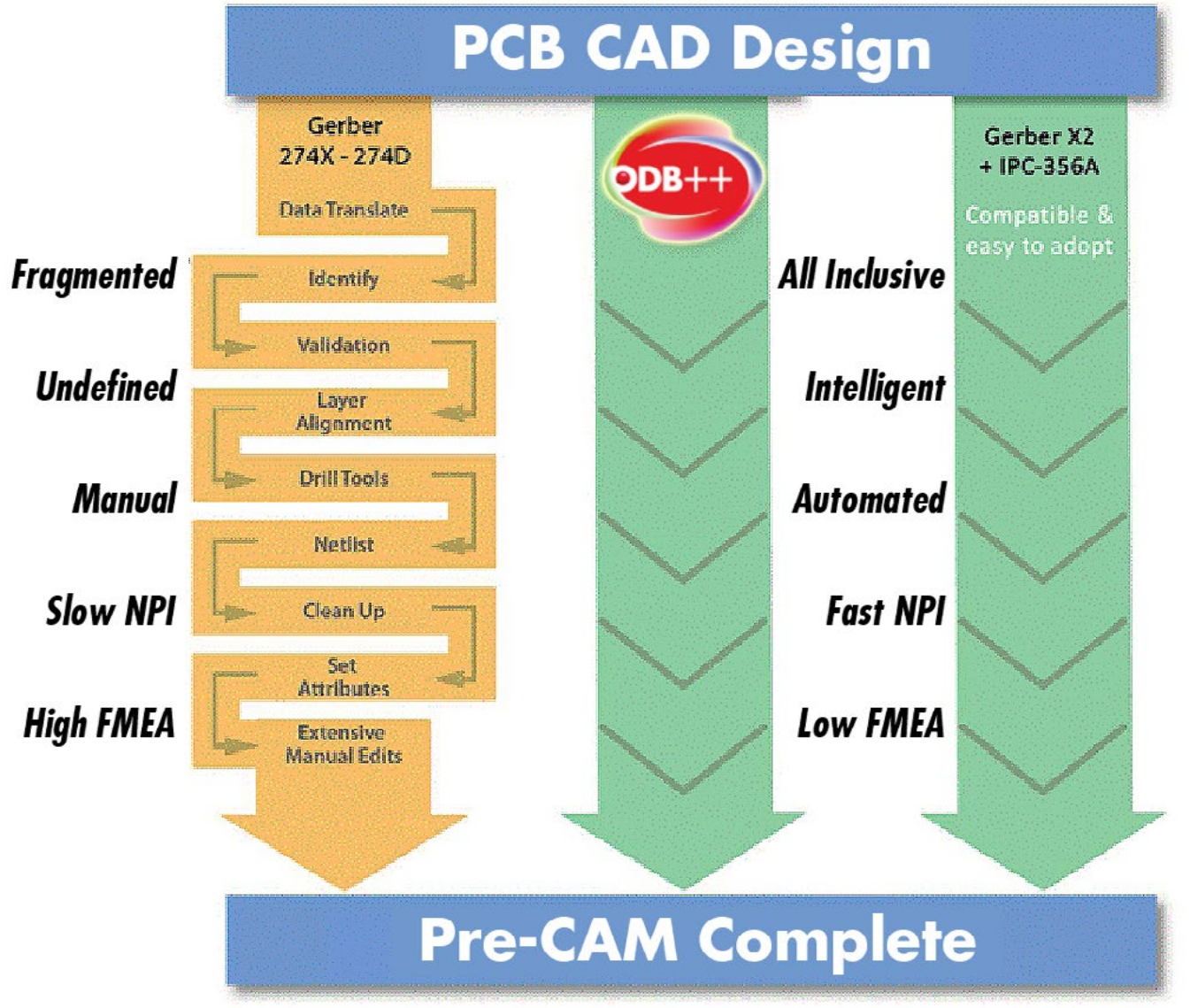


Figure 1: The column on the right can be achieved at low cost, without breaking workflows, in an upwardly compatible way.

way forward is to use Gerber for the graphical data and another format for all the other information that Gerber cannot carry. Thus, he promotes the idea of intelligent, all-encompassing formats for carrying data, but excluding the graphical part. Why reject the advantage of having all of that other information linked to the graphical objects as well, and vice-versa? The problem that needs solving is taking all of that fragemented data into a single coherent model comprising both the PCB bareboard and the assembled PCB. Keeping parts of the product-model seperate for simplicity is fine if you are only interested in a narrow subset of the PCB product-model, but it is a big problem if you need a complete definition of the product, as do all DFM and NPI engineers! There is no escape from the fact that, sooner rather than later, the data must be integrated.

Reductio ad absurdum: To take the idea to an extreme, maybe there is a drilling expert out there ready to explain that Excellon should be used for holes information, but all of the "other information" (including the layer graphics, no doubt) should be carried in ODB++. Obviously it is absurd to keep part of the PCB product-model (in this case, the holes) separate from all the rest. The first thing a CAM engineer would do in this case would be to read the Excellon file and integrate the hole data into the ODB++—an unecessarily time-wasting and potentially error-prone process.

There is a broad consensus across the industry that fixing the highly fragmented nature of the CAD-to-CAM data files problem is long overdue, and that the answer is to implement integrated, intelligent formats such as ODB++. Many have already taken the step with ODB++, attesting to the benefits of having a more streamlined design-to-manufacturing hand-off process. Over a million different PCB designs have been processed into manufacturing using the ODB++ format since its introduction. It works, and is widely implemented by some of the largest electronics OEMs in the world, as a standard part of their NPI business process.

What limits the implementation of ODB++ more widely? Why do people still use all those fragmented narrow-scope data formats such as Gerber, Excellon, netlist, component-placement list, etc? I would suggest that the reason is not technological; it is a combination of business and human factors. Firstly, it costs money to change a business process; tools have to be upgraded. But in order to gain the time/cost/quality advantages, an investment has to be made, and that is nothing out of the ordinary. Secondly, there is a perception that continuing to use the old method is not only free but also "safe," whereas to use the new method is expensive and "uncertain." The "safe" versus "uncertain" part is the human part. There is an jargon-acronym for it: FUD, which stands for Fear, Uncertainty and Doubt. The same was true when the Gerber format was introduced. Using it required a high level of investment, and it took time for the industry to see that the benefits outweighed the uncertainties even though the idea of it was obviously a good one 50 years ago. Hand-drawn artwork was still used for many years after, even though a better method (Gerber data) was available. It took time for the industry to make the change. But change is inevitable if businesses intend to advance given the complexities of today's systems designs. This is why I advocate ODB++ as the new data format standard.

Julian Coates
Director of Business Development
Valor Division of Mentor Graphics Corporation

Karel Tavernier's Rebuttal:

And the Data Transfer Beat Goes On...

In a recent article, Smart Data Formats Automate CAD/CAM (February 2014), Julian Coates of Mentor Graphics wrote an article about the ODB++ format. My reaction to this, Gerber—the Smartest Way Forward, appeared in the July 2014 edition of the same publication, as did a rebuttal by Coates of my article.

Here I would like to rebut Coates' rebuttal of my rebuttal. To be merciful on readers, I will keep it brief, so that the rebuttal process converges rather than spinning out of control.

In Coates' July rebuttal, he wrote: "No doubt Gerber is a very fine format for defining the graphical layers of a PCB."

That's good. My impression was that Coates saw Gerber as an intrinsically error-prone image format whereas I maintain there are very few errors when transferring images in Gerber. So we both agree that Gerber is a very fine image format. Where our opinions diverge is in how we proceed from this fact. Coates went on to state: "At a recent industry debate, I suggested that the best way forward is to use Gerber for the graphical data and another format for all the other information that Gerber cannot carry."

Thus, he promotes the idea of intelligent, all-encompassing formats for carrying data, but excluding the graphical part. Why reject the advantage of having all of that other information linked to the graphical objects as well, and vice-versa? The problem that needs solving is taking all of that fragmented data into a single coherent model comprising both the PCB bareboard and the assembled PCB.

Actually, in no way do I reject the idea of linking all the other information to the graphics objects. On the contrary: It's clear that a PCB is more than a set of images, and all the data describing it must be transferred as a coherent whole. Here, too, we agree. Where we disagree is how we achieve this coherent whole. Coates believes that the wholesale adoption of ODB++ is a practical way forward. I do not. In another passage from his July rebuttal, Coates correctly analyses why ODB++ is not more widely used:

"What limits the implementation of ODB++ more widely? … I would suggest that the reason is not technological; it is a combination of business and human factors. Firstly, it costs money to change a business process; tools have to be upgraded. […] Secondly, there is a perception that continuing to use the old method is not only free but also safe, whereas to use the new method is expensive and uncertain. The safe versus uncertain part is the human part."

These are entirely rational and justified concerns, and clearly the vast majority of this industry feels that they outweigh the benefits of ODB++ (or of any new format that has been tried over the decades for that matter). Who am I to judge that the whole industry is wrong? That said, our industry must move on, and like Coates, I too would like to see CAD to CAM data transfer advance beyond current practices. This is why I propose a path that is far less expensive and risky than that advocated by Coates.

The first step along this path was to clarify areas in the Gerber format specification that were sometimes misunderstood, and to remove elements in it that were outdated, rarely used, or superfluous. This has been carried out in recent years, so the current spec is clear, sharp and to the point—there are no useless bells and whistles in the Gerber format.

The second step, completed earlier this year, was to introduce the Second Extension, or Gerber X2 format. Gerber X2 contains attributes that specify how the layers stack up, identifies via pads, indicates where the impedance controlled tracks are, and describes a host of other parameters that support the image data. With X2, what was missing in Gerber has now been added—in Coates's terminology, the attributes add intelligence to the format. The neat thing is that they do not affect the image, which means that existing workflows are not broken: X2 requires only minimal changes in working practices, and certainly none that would require approval, testing and all the rest. The fully X2-compatible CAD and CAM software will read entire Gerber X2 archives automatically, with all layers in place, while identifying the function of each object. And even in combination with older software that does not support X2, the correct image is still produced. This means that even if users do not reap the full benefits of Gerber X2, they can happily move within the X2 world without problems. Ben Jordan of Altium concurs: "ODB++ is a good standard, but Gerber X2 does solve the problems while being backwardly compatible."

More importantly, this means that nobody is forced to buy anything, and Gerber users can decide in their own time if, how and when to adopt new X2-ready software to take their processes to the next level. For those interested, there is a sample X2 job on the Ucamco download page. It shows the simplicity of the concept. Download it and try it on your own Gerber input software—in all probability you will be able to read in the images correctly, but your software will throw some warnings. This demonstrates the compatibility of X2 with non-supporting software.

THE GREAT GERBER VS. ODB++ DEBATE *continues*

Coates makes much of the claim that ODB++ is a single format and that what I propose is a collection of different formats. This underpins his argument that the good old Gerber format should be dumped and replaced with something entirely new. This is a curious argument indeed: ODB++ is in reality a collection of folders with different syntaxes for each type of data, which are all zipped together in a single archive file. In my opinion this is not a showstopper—on the contrary, it's an inevitable consequence of the fact that components, materials, graphics and netlists are all entirely dissimilar objects. They must all be stored in appropriate formats, each of which, by its very nature, is very different from the others. All they have in common is the ODB++ name. This is clearly demonstrated by the following: if ODB++ image input is implemented in your software, it will not miraculously read materials. Even though you may be able to write images, you cannot automatically write a netlist. These are separate items that require individual implementation, each with its own specification, each in its own folder. This is OK; it is impossible to put these intrinsically different objects into the same format. But I fail to see the difference between zipping together a collection of very different ODB++ folders, and zipping together Gerber and IPC-D-356A files. To anyone who might object that 356 is a different format from the Gerber format, I would propose the following thought experiment: Take the 356A specification, tear off the title page and replace it with a page with the title "Gerber Netlist Format." Lo and behold, now, images and netlist are in the same Gerber format! In other words, there's no substance to the claim that ODB++ is a single format—it's all in the name. Of course, in both cases, the information must be consistent. If you offset the netlist to the image, well, you have a problem, both with Gerber and ODB++.

What I propose is that we, as an industry, take a practical and pragmatic route to improvement: by keeping what works well, changing what does not and adding what is lacking. With Gerber X2, we are doing just this, as Graphicode's Paul Wells-Edwards points out: "The beauty of Gerber is that it's simple, and very widely used, and Ucamco's use of attributes is a very clever and straightforward way to improve and build on it. By extending the format and making it far clearer, Ucamco has improved the CAM task no end."

Indeed, it makes no sense whatsoever to totally abandon something as good as Gerber's image format, which covers the most difficult and critical part of any PCB data archive, to resolve issues relating to the archive's far simpler elements. The industry intuitively senses this, and this is why it has stayed with the Gerber format.

X2 has been designed to be easy to implement and easy to adopt, as it consists of just three new straightforward commands, and support for it is growing. Graphicode is pioneering the X2 wave with GC Prevue v22.3, the industry's first X2-ready viewer software, which is now available for download. Altium too has been quick to recognise the value of X2 and will support it in the upcoming version of Altium Designer. I was recently informed that DipTrace and Kicad will also output X2 in the course of 2014, and LPKF will support it from Q1 2015. Eurocircuits and AT&S offered to beta test it, and it will be in real production by the end of 2014—less than 12 months after its introduction. This demonstrates the benefits of smart improvements: fixing what is broken but leaving in place what works well, which takes into account the community's legitimate concerns about cost and risk.

This is why I advocate X2 as the smart way forward.

Karel Tavernier
Ucamco