TEX and Reality
Clashing Mindsets?
There can be several reasons for using \TeX. Some are subjective. You like the way it works: you edit a document using a simple editor, add a couple of directives and delay rendering. It's the content and structure that matter. You need it for instance because you have to typeset math and you believe that no other tool can do a better job on that. You found out that it can save time because it is programmable and after all, programming is a nice distraction from writing. You don't want to change a 20 year old habit and why quit using something that you know well by now. You like an occasional fight with a batch oriented system and updating (sometimes to the extent of compiling) can be done while watching a movie. You dislike learning a new program every 5 years. Of course it would be different if we'd live for 500 years.
I've always used TEX and can do what I need to do with it. I like to focus on what can be done instead of what can't.

I don't like disposable tools and am quite lucky that T EX still can adapt to my needs.

I like my job but only when using the current tools and cooking up reuseable solutions.

I need it for rendering (often educational) content and also use it for fun.

In the process I need to implement styles based on designs provided by designers, most probably only know click and point tools but some of them can think outside that box.
Each project has at least a few challenges, the input, the design, graphics, the boundary conditions, interfaces, etc. In quite some cases a printed product is an afterthought and coding is driven by viewing on the web. Most time goes into mapping structure. Coding is done in xml because we can then manipulate content and publishers can reuse it. Publishers often use a preselected designer and ask him/her to come up with a design. Chapter openings and title pages take some effort as well, especially if the implementation has to be exact. For some reason design comes before content so the designer has to guess. Although one can try to catch bordercases it hardly pays off as the eventual solutions are not that logic. Simplification is preferred over heuristics. Unfortunately designers never use the fact that we can program variations and flexible solutions. On the other hand in a later stage we can quite conveniently provide solutions for problems resulting in the editorial workflow.
Structure in regular TEX documents assumes a proper nesting of chapters, sections, subsections etc. In ConTEXt we can clone heads and configure them independently. Often we end up with tens of variants. In practice numberings can intermix, for instance subsections can be numbered per chapter instead of per subsection. Numbers seldom run like 1 . . . 1.1 . . . 1.1.1 and individual components can be omitted and can have different properties (font, color). This quickly becomes messy as more (unexpected) structure is added.

It's for this reason that we now have a more complex model of resetting and synchronization of states in ConTEXt. Actually we keep adding more structure support. Additional information that is used in a chapter sometimes is also used elsewhere, as in tables of contents (for instance icons). Therefore in ConT EXt MkIV we now have the possibility to let userdata travel around.
When making a product line it helps if there is some systematic approach in defining colors but it does not work out that way. Unfortunately we can never use the colorpalet and colorgroup features that have been present in ConTEXt from the start. Spotcolors are nice as they enforce a more systematic approach than process colors. In such cases there is often some system. With processcolors we often have to fight the "on my screen 0.01 % makes a big difference" dilemma. Automatically converting graphics to such color spaces can save a lot of time and money.
Although there is some fashion in using fonts most designs use at least a few different ones. Not all fonts are equally well equipped and one cannot rely too much on features without testing them first. Although OpenType makes things easier it also introduces problems due to incomplete features.

A macro package assumes some logic in sizes and relations but this is of no use in practice. Most of the font mechanism is simply not used. The same is true for interline spacing. Often some standard Latin quote and title is used to determine the spec. Not seldom most spacing is inconsistent.

It looks like justification is not wanted that much, let alone advanced features like protrusion and expansion. Inter-character spacing is sometimes requested.
We started making ConT EXt for our own use, especially complex and demanding educational documents. Nowadays we stick to typesetting and as we specialize in automated processing we have to operate within strict bounds. We use not that many handy features as there is hardly any structure in the designs we have to implement. But we use quite some of the manipulative power of ConT EXt. Also, we are able to fulfil even the most extreme demands. It's user demand that is the driving force behind most new features. Users typically use ConTEXt in a different way than we do. And . . . some things can probably only be done with T EX, especially in automated workflows.
Talk to those implementing the design, let them show you what can be done. Stick to general designs and don’t go into much detail. It’s the look and feel that matters.

Think in systematic solutions. Lack of freedom in interactive placement of graphics can be compensated by other variations.

Think outside the box. Use the fact that the system is programmable and can adapt. And it probably goes beyond what you can think of.

Try to make a design extensible. There will always be more structure. Some components will have less text that expected. Titles can be very short or quite long. Keep in mind that you cannot tweak.

Try to see a pattern in structure and provide escapes for strange cases. Give the implementor some freedom.