10 Tips to Architect Your Sitecore Implementation Like a Pro

ENGAGENCY.COM

Website | Schedule a Meeting

So, you want to build a Sitecore website.

Your CTO signed off on the licensing costs, your CMO has big plans for all the revenue it's going to bring in, and your marketers can't wait for the new functionality they'll get to use. Now all you have to do is actually build it.

Architecting a Sitecore implementation is complex. You're not only laying the foundation for your site today, but for everything else you hope to achieve with it in the future.

Here are our top tips for architecting your Sitecore implementation like a pro, based on our own experience of doing this for well over a decade.

1. Follow Helix principles.

Step one to a successful Sitecore architecture is following Helix principles. Sitecore Helix is a set of official best practices for implementing Sitecore, outlined by Sitecore themselves. It doesn't get more expert than that.

Following these guidelines is critical to ensuring your Sitecore implementation is architected in a way that allows you to build it efficiently up-front, and that it remains easy to use, maintain, and troubleshoot moving forward.

Do not make the mistake of ignoring Helix when creating your Sitecore architecture. This is critically important!

2. Map everything out first; then architect it.

It is essential that you map out *all* of the components of your project *before* you get lost in the details.

We're including this step near the beginning because it's extremely important to get this right. In order to architect an elegant and logical solution, you need to have a deep understanding of the various complex relationships between the data.

At Engagency, we typically prepare a spreadsheet that maps out all of the parts and pieces. This includes, but is not limited to:

- How the content will be organized in Sitecore
- What belongs in a module vs. what belongs in a page
- What building blocks are needed, including all data templates, page renderings, module renderings, and layouts
- What supporting information will be needed, such as category lists or color themes, and how can these best be generalized for maximum usage

3. Be thoughtful of your Content Editors.

When you're architecting a Sitecore implementation, keeping in mind the needs of your visitors and customers is obvious. However, you also don't want to lose sight of the needs of the people who will actually be using Sitecore to manage the website. That includes your internal team, such as the content editors and marketers.

A good Sitecore architecture makes page creation and editing a simple, straightforward process.

Your Sitecore partner's job is to determine the best way to configure the CMS so that modules, page templates, and personalization features are easy for your content editors to find and edit as needed.

4. Consider content reusability.

For your marketers to regularly churn out revenue-boosting content for your organization, they need to be able to do so efficiently. Making your Sitecore content reusable is a big part of making that happen.

If your content will be used across multiple pages, or multiple sites, your Sitecore partner must architect a solution that makes it easy to store and reuse content wherever it's needed.

5. Think in terms of whole modules; don't get too granular.

At Engagency, there's one thing we often see done poorly, when our clients bring us in to fix a bad Sitecore implementation that's been architected by a previous partner. This all-too-common, misguided approach is to take every little thing and make a module out of it.

The end result is a Sitecore solution overloaded with modules. You'll have too many for your marketers to keep track of or know what to do with. This makes it extremely difficult and frustrating for your marketers to do their job, and even harder for your developers to troubleshoot things when something goes wrong. It's the Sitecore equivalent of finding a needle in a haystack.

Here's what we recommend instead. Consider all the elements that make up part of a content piece, and turn them into a module.

For the sake of example, let's use a customer testimonial. A testimonial typically includes a header introducing it, a copy section that includes the actual testimonial, a photo of the person who gave the testimonial, and maybe a smaller copy section identifying that individual and their company. All of these elements can be tied together in a single module that's reusable *across* your site, rather than forcing a content editor to makeshift their own testimonial module by selecting a header module, copy module, and photo module.

6. Don't forget about personalization.

The piecemeal approach we advised against in the previous tip becomes even more problematic when you consider personalization. Personalization is one of the most powerful features offered by Sitecore, and when you get too granular with your modules, you end up shooting yourself in the foot.

Typically, you want to personalize a whole module in Sitecore. For instance, you may want certain testimonials to appear on different page types, based on where that traffic is coming from. If you have a partner who sends you a ton of referring traffic, you might want the testimonials on your site to include any you have from that referring partner.

If you took the super granular approach, your marketers will need to create personalization rules for all the individual elements in that testimonial, rather than applying it once to the module as a whole.

You can avoid this nightmare by making sure your Sitecore partner is thinking about personalization *throughout* the architecting phase. You should expect your partner to create a thorough architecture document and guide you through any modules or page renderings they're creating.

7. Distinguish page content vs. module content.

We've talked about tying elements together into a single module. Now let's discuss how you can tie modules to pages in Sitecore.

It's likely that many of your pages will fall into certain categories. You'll have a preferred format for your blog articles, your product pages, your long-form resource content, and so on.

While your marketers will add unique content to each of these pages, and there will be elements that will be personalized on each of them, it's likely that they'll share some default elements. A blog page, for example, may have a feature image, a title, a copy section, and perhaps a CTA for people to download a whitepaper or sign up for your newsletter. There may also be a right rail to the page, where you feature related articles, trending articles, or buttons to follow you on social media.

Make things easy for your marketers by pre-loading these modules *into* the page template. If every page of a certain page type will contain the same module (like the social buttons on the right), it should be handled at the page level, *not* the module level.

8. Identify global vs. page components.

Along the same lines, if a component is global, it shouldn't be explicitly assigned to a page. Rather, it should be part of the site structure.

For example, your Content Editors shouldn't have to take action to make a header and footer display everytime they publish a new blog post. Those global elements should simply show up as part of the layout or background coding.

Otherwise, your Sitecore partner is setting you up for failure by allowing your website to lack consistency. That not only looks unprofessional, but it often means important things get left out, too (like the phone number your customers use to call you).

9. Transform wireframes into patterns.

Now it's time to tap into your inner Neo and pretend you're in *The Matrix*.

If you're working with a design team, make sure your Sitecore development team is carefully reviewing their wireframes with an eye for re-usability. For example, your product page may include testimonials, while your blog articles feature quotes from experts. These may appear as two different things, but given the similarity of the content, making a single module to handle both use cases would be expeditious.

This is where it's helpful to step back and remind yourself that Sitecore is both powerful *and* flexible. A module can handle different things. Think more about the *form* of the content vs. what's in it. With the example we just mentioned, you can use the same module across both page types, rather than having two different modules (a *Testimonial* module and a *Quote* module) that include the exact same elements.

In Sitecore, you could even choose to make certain elements of this module optional for display. If your Content Editors have a featured quote they'd like to call out in a blog, they can use this Testimonial module, and just check or uncheck a box for whether they want the photo portion to display.

So that's step one. Step two is naming your modules in a way that Content Editors can understand them. For this, we recommend again looking to the form of the module, vs. the content within it. A testimonial module that doubles as a blog quote module may be better named as a "Quote" module. You want to find the balance between having a billion granular modules and making it easy for your content editors to quickly find what they need.

10. Prep your Sitecore architecture for presentation.

Finally, it's time to present your Sitecore architecture to the stakeholders for approval!

At Engagency, our practice is to explain our Architecture from a bird's-eye view and then break it down into individual components. First, we walk through how all of the components work together to create a cohesive whole. Then, we review the details of every item needed, where each is stored, and how each item works with other components. While we are at it, we take the time to define how Sitecore works, explaining the difference between a rendering vs. a module vs. a global element.

If your Sitecore partner is not willing to go to these lengths to ensure you have a clear idea about what to expect, then you might want to consider looking at other options.

Want to see what one of our Sitecore Architecture Documents looks like? Contact us here and we'll share an example with you.

Creating your Sitecore architecture

As you can see, architecting a solid Sitecore implementation is a complex process. Not only that, but the steps you take during the architecting phase have ripple effects that can impact the ultimate usability and performance of your solution in the long run.

Sitecore architecture is not something you want to leave to a design agency. To do it right, you need a team of Sitecore experts who know how to collaborate with your design agency and translate their wireframes into a well-built Sitecore solution.

At Engagency, our expert Sitecore Architects ensure your implementation is built in a way that serves the needs of your users, your marketing team, *and* your developers. Your users will love interacting with your site, your marketers will love managing your site, and your developers will love how easy it is to maintain, enhance, and scale.

For those of you interested in architecting your own solution, we suggest our Sitecore Co-Development & Enablement Services. Here, our Sitecore Architects will lend their decade-plus of experience as they work with your developers side-by-side, architecting a solution that performs well and is pain-free to maintain.

Contact us today to learn more.