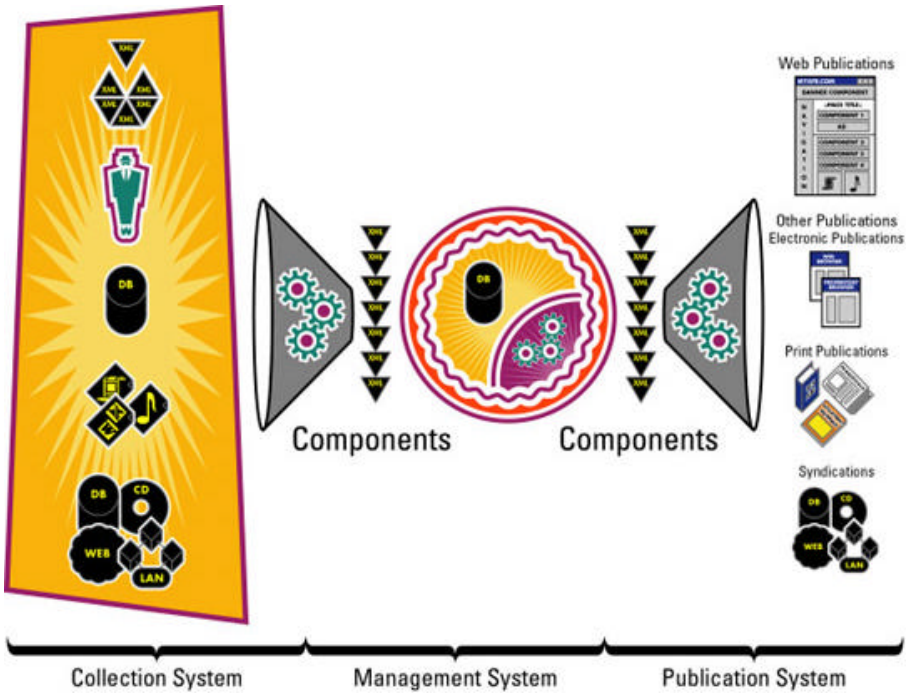
Doing Requirements and Logical Design

A CM Domain White Paper

By Bob Boiko



This white paper is produced from the Content Management Domain which features the full text of the book "Content Management Bible," by Bob Boiko. Owners of the book may access the CM Domain at www.metatorial.com.

Table of Contents

After you have secured a project mandate, you can begin to collect all of the information that you need to design the system, with your sponsors and other stakeholders in mind. You can begin by gathering your organization's requirements for content, publications, and CMS infrastructure. From there, you can create a product-independent (or logical) design for your CMS that defines exactly how you intend to collect, manage, and publish information.

In this white paper I'll lay out some project techniques and deliverables you can use to collect and organize requirements and construct a logical design.

Note

This white papercovers the project process behind requirements and design. The actual requirements and design considerations that you need are covered in the chapters in Part IV, "The Logical Design of a CMS."

What Are Requirements?

Requirements are the specific qualities, features, and objects that the system will ultimately include. In a CMS project, there are three sorts of requirements to be collected:

- **Content requirements** which specify the kinds of content to be managed and how it must be gathered and organized
- **Publication requirements** which specify the kinds and structure of outputs you want the CMS to produce, including how you want to target content to your chosen audiences
- **CMS requirements** which specify how the CMS hardware and software are required to operate

It is likely that you already have a lot of this sort of information hanging around. By the end of this stage of your project, however, you need to have these requirements fully nailed down.

I think the word "requirements" says it all. You will be required by your organization to follow these guidelines in designing and constructing your system. If you take this seriously, it can work very well for you. In particular, you will be very careful about what you call a requirement. You will be sure that 1) the statement is of enough importance to be called a requirement, and 2) you are happy to be held to it. In addition, as you collect and organize requirements, you can get the organization to help you by underscoring the importance and durability of requirements.

What Is Logical Design?

At the end of requirements gathering, you will have a head (and a hard drive) full of other people's dictates to you about how the CMS should be, and why it is being, created. Next, it is your turn to respond. In logical design, you decide how the CMS will meet all of the goals and requirements that have been set out for it.

I use the term logical design to contrast this process with the later process of physical design. In physical design, you will put together the specifications and plans that say how the CMS will

actually be constructed from hardware and software. In logical design, you put together the system outside the constraints of a particular platform. Another term for logical design might be platform-independent design. So, why would you need to do such a thing? Here are a few good reasons:

- There is much more to a CMS than hardware and software. Content, for example, is neither hardware nor software.
- You should understand what you *want* to happen before you decide *how* you will make it happen. There is nothing more time-consuming and potentially career adverse than creating a system, only to find out later that it can't do what you never took the time to figure out that you wanted it to do.
- You need information to help you decide what sort of hardware and software you will need to build your CMS. The best tool you have for deciding what sort of platform you need is to know what you want it to do. Even if you already have a CMS in place, before you start pushing a bunch of buttons, you should have a clear idea of the results you intend.

Logical design is like a big puzzle. The pieces are bits of information you collect or derive from the requirements related to authors, sources, components, publications, and so on. You have to fit them all together so that they make a complete and sensible picture of the collection, management, and publishing process you intend to create. If a piece does not fit, you must rearrange the others until it does, or get rid of the outlier.

Logical design results in process, relationship, and structure, as follows:

- **You design the processes** by which you intend to collect, manage, and publish content. For example, you must figure out how a particular kind of content can be collected at the right rate to meet the publication cycles of all the publications in which it will appear.
- **You work through the relationships** between the significant players in the system until they all support and enhance each other. For example, you must figure out how you can get your authors to create content components in such a way that they can be targeted to your various audiences, using publication templates.
- **You design a content structure** that fully represents the information you need to distribute, and adds enough metadata to it to enable you to automate its management. For example, you need to create a set of content component classes that organize the information you want to deliver, and augment it with management data that you can use to automate its delivery.

By the end of the process, you might long for the days when everyone was telling you what to do. Or, if you're like me, this might be your favorite part - figuring and finessing and finding a way to make very human factors fit into a logical and very effective framework.

Techniques for Getting the Job Done

The techniques in the following sections help you understand what I mean by requirements and how you might go about gathering them, including how to approach your organization most effectively to get them to give you the requirements you need. They also help you learn and practice some techniques for logical design, including how to go from business goals to a system, how to iterate a number of times through the process, how to triangulate on the various aspects of design, and how to manage the enormous amount of detail you will collect.

Gathering requirements

The requirements process I will lay out is quite a bit simpler than the one most software developers use. Rather than asking an exhaustive set of questions that will lead directly to a

design, I recommend that you ask simple questions, which provide you with the overall direction you will need, in order to dig for the more fundamental questions and answers.

The process I present is predicated on the assumption that the people you will approach for requirements will have neither the time nor the insight to give you real answers. They do, however, have the ability and the authority to determine the broad outlines of the system.

The requirements process

Your requirements process needs to result in an agreement on content, publication, and system requirements. The point here is to get the people in the organization to tell you what they really need. You are not trying to get them to learn all of the terminology and process that you will need to turn requirements into action. Nor are you trying to get them to work any harder than absolutely necessary. So, your best approach, I believe, is to ask a set of simple, yet comprehensive, questions that require only short answers.

Before diving into the more precise set of questions to ask, let me propose an overall process for gathering requirements: canvass widely, catalog everything, and get consent.

Canvass widely

Cast a net as wide as is feasible to get input from a wide base of contributors. With any luck, through your previous work in assessing the organization and building a mandate, you have already isolated the list of potential contributors. The list ought to be narrower than the one you used originally to find out what was going on in the organization. It should include only those people who are in a position to "require" something of the system. On the other hand, the list should be wider than the people who participated in the mandate process. They are probably not the only ones with the knowledge and authority to state what the CMS must accomplish.

Tip

You can certainly get the process off to a good start by providing a list of the potential requirements that you already have amassed from your initial communication. It also helps to ask different people for different types of requirements, based on what they are qualified to comment on.

Catalog everything

Your position in the requirements process is scribe and librarian. The most help you can offer is to keep the list of potential requirements from getting out of control and becoming confusing. The work you do to synthesize people's input into an easy-to-read and easy-to-understand document is crucial. When someone does not see her words in the document you produce, you should make it clear that her requirements are being superseded by larger and more central concerns, rather than ignored. There is a subtle art to combining requirements in a way that no one feels slighted, but that inferior input is weighted less than superior input.

Tip

A technique I have found useful here is not to wait until I am finished to get feedback. Rather, I go back and forth with a number of contributors, passing small pieces of the requirements between us as I work on the final draft. This technique has the effect of giving work to people in bite-sized chunks (that is, the size of chunk that can get done in an e-mail exchange) and building consensus before the draft is even circulated.

Get consent

Formal approval of the requirements is a must. When someone comes along weeks (or even months) later with a "must-have" requirement, you need to be able to point to the formally approved requirements. This may not stop you from having to include the rogue requirement, but

at least it will make the "rogue requiree" think twice. In addition, the formality of official consent helps people take their part seriously. It may be that by now people are tired of all the work that they have had to do to get this project started. You can assure them that this is the last big push for them, and smile inwardly that you have exhausted them into giving you the space you need to do the tasks ahead.

In addition to approval on the list of requirements, you should push for consent on their priorities. Unless I am sorely misinformed, your list of requirements will be longer than you are comfortable accomplishing in the limited time you have for the project. In addition, it is likely that the list you have includes items that are there because you stopped arguing or simply because someone was willing to stand up for them. By having priorities, you can let the organization decide the relative importance of the requirements and prepare the organization for not getting all required items on the first go-round.

Tip

You can get your contributors to take the first cut at prioritization by asking them to mark requirements as "must have," "should have," or "could have."

In gathering requirements, your goal is to get people to think out-of-the-box to tell you what is really needed, not just what they already have or are already doing. You want people to give you the constraints without which the CMS will not work, rather than tell you about their favorite white papers, Web sites, and development platforms. There is a fine and not-so-visible line between a desire and a constraint (especially if the "desiree" is a person with sway!). If there are disputes in the organization, especially in the platform questions, they will show up in your requirements. If you are lucky, the support and consensus that you forged in the mandate process can carry you through any difficulties here - but don't count on it.

Note

I have seen more than one content management initiative stall as all of the organization's disputes were visited on the content management requirements-gathering process. A software developer I once knew captured the problem with requirements and the organization most elegantly. He said, "When you pour milk on oatmeal, all of the lumps come to the surface."

Content requirements

For clarity, I will start with content requirements. Content is a good place for you to start if your organization has its business goals solidified. If not, you may get stuck by starting with content. Without a solid marketing plan, you may not know who your customers or other audiences are, or which are most important, making it extremely difficult to define and prioritize content. Without clear business goals, the simple question, "What content should we deliver?" may open a can of worms. The result may be a CMS effort that either forces those plans to be solidified, protracting the process, or an effort that pushes on without clear goals and ends up with a half-baked system.

If you find that you are not making much headway with content, try starting with the questions about publications and working back to content.

The questions behind the content requirements are quite simple:

- What information do we need to deliver?
- For each type of information, who in the organization is needed to create this information or how will we otherwise acquire it?

From these humble beginnings, you can begin to construct the entire collection system. The answers to the second question serve as much to make the respondent think as they do to provide you with information. If the respondent has a clear idea of how you will come by the information she suggests, then great. If she does not, then maybe question two will cause her to

think a bit more before suggesting that you collect such information. You can also use the answers to question two as an aid to prioritization. If no one knows how to get a hold of a certain kind of information, then maybe the cost of collecting it is higher than the value that the content provides.

System requirements

System requirements are the set of technological constraints that the organization would like to put on the CMS.

There are a few more questions to ask to gather system requirements, and they are more specific than the content and publication questions. Following are such types of questions:

- **System integration:** With what existing systems will the CMS have to integrate? Existing systems include those connected to the management system, as well as those that provide information to the publishing system.
- **Hardware and software:** What company standards for hardware and software apply to the CMS? If, for example, all of your graphic artists use Macintosh computers, then you need to know that so you can make sure your collection software works on their computers.
- **Development environments:** What requirement or preference does the organization have for software development environments? The question is not "What platform do you prefer?" but, rather, "Has the organization or any department mandated the use of a particular platform?" This phrasing helps avoid getting back a lot of personal preferences (which, for some strange reason, you do not need any great process to find out).
- **Deployment platforms:** What requirement or preference does the organization have for publication deployment platforms? The obvious answers are about Web servers, Web application servers, databases, firewalls, and server farms. You will find this sort of information easiest to collect. Either the organization has all of that set up and prefers to continue using it, or it does not and it will be part of your job to figure all that out. Less obvious are the deployment platforms for non-Web publications. For example, what infrastructure exists, or is preferred, for print publications? Is there a group that really has this nailed down? What e-mail platforms are in use that can do bulk mailings (if that is part of your publication set)?
- **Performance:** What requirements are there for scaling and performance of the management or publication systems? What you are most likely to get are the current statistics for your organization's Web initiatives. What you might probe a bit deeper for, though, is how far the current systems can go before they begin to degrade. It is quite likely that you will push your systems much harder with a CMS behind them than they are being pushed now.
- **Deployment:** Are there standard rollout procedures? If you are in a large organization that has existed for more than a year or two, then it is beyond doubt that your organization has rolled out an enterprise system before. If this rollout was done well, it will have left a written legacy of good procedures and guidelines. If you can find this legacy, then you will save a tremendous amount of time and effort.
- **Maintenance:** Are there system maintenance standards? Your organization probably maintains enterprise systems now. What standards are used to measure the quality of maintenance agreements and support procedures? As with rollout standards, you can save a lot of time if you can find an existing process into which you can tie the CMS.
- **Localization:** Are there localization standards? If your organization is in more than one culture, you have no doubt already come across the issues associated with multiple languages and localization of content (localization includes, for example, making sure half-naked bodies do not show up on screens in cultures where they offend). I have seen few organizations that have dealt with this problem in a comprehensive and well-documented

way. If your organization has done so, in addition to having a great career opportunity in localization, you will have a great start on this difficult issue.

Publication requirements

The questions to ask about publications are a bit more complex, but ought not to cause any huge workload for the respondents. They are the following:

- Who are the top three audiences we are trying to serve with our information?
- What categories do the top three audiences fall into?
- What other audiences would it be nice to serve?
- How do we provide information to the top three audiences now?
- How would these audiences prefer to have information provided?

By starting with the audiences, you put the respondent in the right frame of mind. Rather than telling you directly what publications they think the organization needs to create, they tell you how audiences prefer to get information. The questions nudge the respondent to begin to prioritize audiences and set them all at the same level. All respondents choose three audiences (an arbitrary low number that you might want to change). That means that they will all have to choose the most important ones to list. In addition, by asking them to choose three audiences, you force the respondents to give you audiences that are at the same level. One can't (or at least has no excuse to) say, "sales engineers, sales reps, and inside sales," while another says, "staff, partners, and customers."

In your questioning process, there is purposely a slot for people to tell you what you do now, but no slot for them to tell you what they think you should do. Rather, there is a slot for saying what the audiences would like. Too often, people will try to fit the audiences into their preconceived notions (or, more to the point, their preexisting outputs) of the right set of publications. This questioning is a not-so-subtle attempt to get people to break those molds.

As with the content requirements, these questions are designed to make the respondents think as well as simply provide information.

Approaching the organization

Sometimes the organization itself can stand in the way of completing your requirements gathering in the following ways:

- **It's difficult to establish ownership of many of the requirements.** Who is the right person to say which database you must use? Who can decide whether a particular type of content is a "must have" or a "nice to have?"
- **People don't have time to answer your queries.** This is especially true for questions that require real effort such as, "Can you sample those 1,000 files and tell me how consistently formatted they are?"
- **People will speak with authority when they have none.** When you ask a question, you may get an answer whether or not the respondent has the knowledge or position to properly answer.
- **People cannot (or will not) deal with your collection methods.** My teams have tried everything from Web-based forms and e-mail messages, to Microsoft Word templates and personal phone calls, and still people cannot understand how to respond in the appropriate way. The good angel on my right shoulder says, "It's not easy enough." The bad angel on my left shoulder says, "Lazy #$%#!"

- **People don't know how to answer your questions,** so they will stall or try to divert you by heaping information on you (such as those people who say, "I don't know what is important so I'll just give you all of it and let you decide.")
- **CMS requirements tend to cross department lines,** so you end up chasing facts around all of the loops and down all of the blind alleys that your organization might have in its structure (for example, being bounced around from person to person until you are back at the person where you started).

So, what do you do? First, a small amount of common sense and organizational savvy can go a long way. Note when you are getting heaped on or put off, and then stop wasting effort. Use your mandate and the support of your sponsors to cut red tape and hold people to the responsibilities of their job descriptions. Note whose interests are being served by holding you back or stalling the process, and act on that rather than on the frustration you feel toward a particular person.

More generally, you can create a plan of attack that identifies organizational contacts and their responsibilities (the plan of attack deliverable is detailed later in this white paper.

Above all, realize that people are unlikely to understand what you are doing and why you are asking the questions you are. With patience, an attitude of respect, a desire to teach them as well as learn from them, and a good sense of humor, you can build alliances while you collect requirements that will last you through the rest of the project.

If you have built your readiness assessment as I outlined, then you have no doubt already interacted with many of the people to whom you are turning now. If you performed a mandate process anything like the one I outlined, then you will have all the organizational support and muscle you need to get back good, timely information.

Finally, use your mandate as a tool to set the context of the discussion. Rather than saying, "What content should we include?" say, "What content best fits the mandate we received?" Keep referring back to the mandate until your respondent (and you, for that matter) cannot help but take it seriously into account.

If all of this good advice fails you and you are still left on Thursday night with not enough information for the Friday deadline, take heart. I have been there, everyone I know has been there, and now you are there too! Welcome to the Frustrated Analysts Club!

Doing logical design

Logical design is the design of the CMS outside of the particular hardware and software you will use to implement the CMS. This is not to say that it cannot help you choose the right system or that it will not be used by the system you end up installing. It surely can, and will. A logical design specifies what is to be done, and a particular CMS dictates how it is to be done.

Here are the aspects of logical design that you will have to take into account:

- **Audience analysis:** This details the kinds of people you will serve.
- **Publication design:** Here you define what content and navigation each publication will include and exactly how they will automatically be built and personalized by the CMS.
- **Component design:** Here you determine the complete set of content components you will manage and exactly how each one will be constructed.
- **Author analysis:** Here you figure out whom you need as content authors and how you will serve them.
- **Source analysis:** Here you decide where you will acquire information and how you will process it to make it ready for the CMS.

- **Access structure design:** Here you define the hierarchies and other access structures that you will need to keep your content organized in its repository and to produce the navigation that you will need in publications.
- **Workflow and staffing design:** These specify the kinds and numbers of jobs and tasks that will be needed to start up and run the CMS.

These various analyses and designs do not stand alone. Rather, they overlap and inform each other in a large number of ways.

Why do logical design?

Most people underestimate the amount of information you need to accumulate to fully define a large CMS. In fact, many organizations skip the stage of defining their requirements for the CMS entirely, and simply begin using whatever software they have purchased, trying to make it produce pages like the ones they have now.

Contrary to what you might expect me to say, I don't believe that this is a bad approach. Given that you have a long time to play with a system, experiment with what it can do, and determine what it requires of your organization, this method can give you a very good start. It can teach you all the basics of content management and give you practical experience with a real tool.

On the other hand, there is a big leap from a system that can produce some pages like the ones you have now, to one that can organize your entire production and distribution process. You can't expect the latter process just to happen by turning on, and using, even the best CMS.

Other organizations rush to buy a tool and then begin an analysis. Again, this can work. By immediately imposing the constraints of a particular product on the project process, you get to use a lot of the methods and tools that the CMS product company might provide. In addition, you can show immediate results and hit a big milestone quickly. On the other hand, if you buy before you study, you risk the very real possibility of having purchased a (very expensive) system, only to find after using it for a while that it was the wrong one for your needs. Finally, by choosing a tool before you really know what you want to do, you may end up not doing something that the tool does not readily do simply because it is off the tool's radar screen (regardless of how much value it can have for your organization).

If you have followed the steps I have laid out so far, you already have spent a lot of time working through the issues and should have a lot of detail about what people want and require of the system. You might think that what you collected during the readiness assessment and mandate process ought to be enough to define the system. Maybe it is, but probably it is not. What you have collected so far certainly establishes what people want, but does it define what your organization needs? At the very least, you will have to organize and augment these wants and get the organization's departments to agree that they are complete. More likely, you have a good start and a lot of general statements but nothing specific enough on which to base a system design.

If you take the trouble to do a complete logical design, it will help you significantly in the following ways:

- **System selection:** In logical design, you say exactly what you want your CMS to do. If you already have a CMS package in mind, then your planning team can use the design to decide the degree of fit between the selected system and the design. If you have not yet selected a system, then the design will give you a clear picture of the system you want. You can compare your picture to the one presented to you by the product companies you review. You will also be able to ask intelligent questions of the CMS product vendors and not be overly swayed by slick presentations. Finally, if you choose to develop a custom system, the logical design provides great input for your software requirements specification.

- **System implementation:** To implement a CMS, you create a set of specifications. The specifications combine what you want to do with how you will do it. The logical design is what you want to do. The features and functions of the CMS you will build or buy are how you will do it. If you do not get a clear idea of what you want to do, then the features and functions of the CMS you use will decide for you. A logical design puts reality into your specifications by making sure that the features and functions it defines are seen in light of the exact content and publications that you want the system to manage.
- **System rollout and maintenance:** The logical design should provide just the right starting point for many of your system deployment deliverables. For example, the logical design describes in detail all of the staff that will contribute to the CMS. The system must be deployed to these same people. In addition, much of the documentation you will need to train users and maintain the system can come directly from the logical design deliverables.

From business to system

You can use these goals and requirements to make your way into a design. As shown in Figure 1, the hierarchy of business goals - a mandate statement is at the tip of an organized set of business goals; a content domain statement is at the same tip of an organized set of publications and content components.

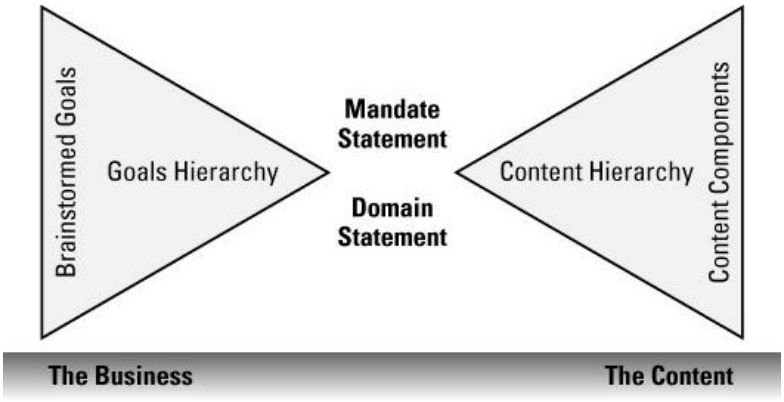


***Figure 1**: The hierarchy of business goals*

Your mandate ought to lend itself to rephrasing as a content domain statement by asking the question, "What information and functionality do we need to deliver to meet this aim?" Then, from the single domain statement, you should be able to start constructing an outline of the kinds of content that belong in the domain (the content hierarchy). In this way, you can pivot naturally from the knowledge of the business you have gained in the mandate process to knowledge of the content you will need to provide.

Tip

When you construct a content hierarchy, try not to be overly influenced by what you already know about the content that people plan to deliver. If you can work strictly from the mandate statement, and then the domain statement, you will have an independent means of evaluating what belongs in the system. Inconsistencies between your outline and what people have asked for in their requirements are a good thing. Use them to rework your content hierarchy, your domain statement, and even your mandate statement (if you dare), or to be able to say with confidence why the proposed requirement does not belong.

With a good content hierarchy in hand, you can begin to decide what sort of content components you will need. This is only a beginning, however, because you will work and rework your list of

components throughout the design process. Loosely, each terminal node (the ones with no children) of the hierarchy is a potential component class.

With the preliminary list of component classes, you are ready to begin the real work. From the content you know you have to deliver, and the requirements you have gathered, you can work through the design of the system that you will use to collect, manage, and publish content (shown in Figure 2).

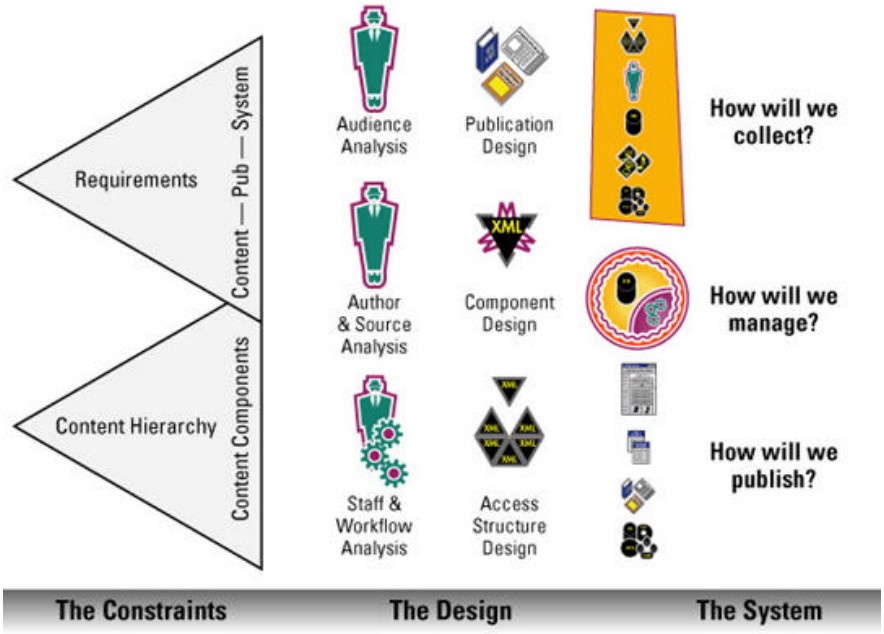


***Figure 2**: Your requirements and the content hierarchy are the starting place for your design.*

Your independently created content hierarchy and the requirements you have gathered are the starting place for design. After you have reconciled these two, you can begin to perform the in-depth logical design of your system. Logical design answers the myriad questions about what you want the CMS to do. You can organize the answers you find into the usual three categories: collect, manage, and publish (for presentation back to the organization).

As you can see, a lot is going on here. You can go from business goals to a preliminary list of component classes, all on a piece of paper in a coffee shop. To go from the preliminary component classes to the final classes and the system you will need for them, however, is another story. You will need to know much more about your audiences, publications, information sources, staffing, workflow, and the access structures you will use to manage components before you can begin to implement the system.

I'm sure you are not surprised to know that there is a lot of work to do. I hope that this overview shows you, however, that the work is neither insurmountable nor unknown. Keep these diagrams in mind (or at hand) as you dive into the details of design.

Iterating through your design

There is no clear dividing line between the amount of categorization you need and the amount you can do. (For example, should you break your audiences into three categories, nine categories, or 27 categories?) Frankly, you can go on forever, collecting and refining your approach to your CMS, and never begin to feel that you have completed the task. In Part IV, "The Logical Design of a CMS," I present a comprehensive list of all the design criteria you should

eventually uncover. If you simply dive in and try to capture all of these constraints at once, you may get bogged down and confused and then run out of time, with an incomplete analysis.

The solution is to pass many times through the same set of constraints, each time going deeper than the last and each time coming to a feeling of completeness at some level. When you run out of time, you may not have all of the constraints worked out, but you will have an even and complete amount of detail in all areas. In other words, approach constraints in a series of deepening iterations. Here is an example of a five-pass process that you can use as a starting place to design your own set of iterations:

- **Pass 1 - Preload:** Review all of the documents and notes you have collected so far and use them to create a general statement for each category of design (audiences, authors, publications, and so on). For example, your analysis so far might lead you to say, "We will have about 30 authors, all of whom will be inside the organization." Use your best judgment to create these statements and note any gaps in your knowledge that cause you to have to guess.
- **Pass 2 - First cut:** From the large number of questions that I provide in Part IV, "The Logical Design of a CMS," decide on no more than three or four questions that you will need answered in each category to feel that you understand the major issues in full, but at a high level. For example, you might divide the 30 authors you came up with in Pass 1 by their savvy and their closeness to your team. Take no more than a few days to get good answers to those questions, and then create a short but comprehensive design report. Get immediate feedback on the report to ensure that you are in synch with the project stakeholders.
- **Pass 3 - Design detail:** Begin to fill in the detailed design constraints in each category (presented in Part IV, "The Logical Design of a CMS") by prioritizing the questions that are most important to get answered most quickly and then acting on the highest-priority questions first. In this way, your details will come together in waves, from most to least importance. You can break this pass into smaller increments, each with its own tight deadline, to further organize your approach. This pass ends when you are at the point of diminishing returns (that is, it is too much effort to drill down any further at this point). You undoubtedly will have important unanswered questions, but you must resolve to push on while you await responses.
- **Pass 4 - Triangulation:** One design constraint does not live in isolation from all others. More often than not, you will discover conflicting constraints or ones that force you to reconsider and, perhaps, deepen your analysis. This is a good thing. It means that you can play one constraint off others to make them all stronger and more consistent. In this pass, you can focus solely on the how the different classes of design constraints affect each other. See the following section for more information on triangulation.
- **Pass 5 - Finalization:** In this pass, you go through the entire set of constraints (which might be quite large by now) to finalize them for approval. It is likely, however, that while you are doing this, the project can begin to move forward into design and specification.

Triangulating on constraints

To find the exact location of a point in the landscape, a surveyor triangulates. First, she measures the compass bearing to the point from one location, and then she measures the compass bearing to the point from a different location. After measuring the distance between her two locations, she can draw a triangle that tells her the exact location of the point in question. If she wants to be more precise, or check herself, then she can use more than two locations. The more locations from which she takes measurements, the more sure she is that she knows where her target is.

To find the exact composition of the content components you must create and manage, you too can look at them from different perspectives(as shown in Figure 3. For example:

- **You can look at your components from the perspective of the collection requirements:** What kinds of components are your authors able to create? What kinds can you acquire?
- **You can look at your components through the lens of management requirements:** How will you determine when it is time to archive or delete a component? What indexes and storage requirements will they have?
- **You can survey your components from the vantage point of your publications:** What kinds of content belong on each page? What sorts do our audiences expect? What will we need to know to personalize?

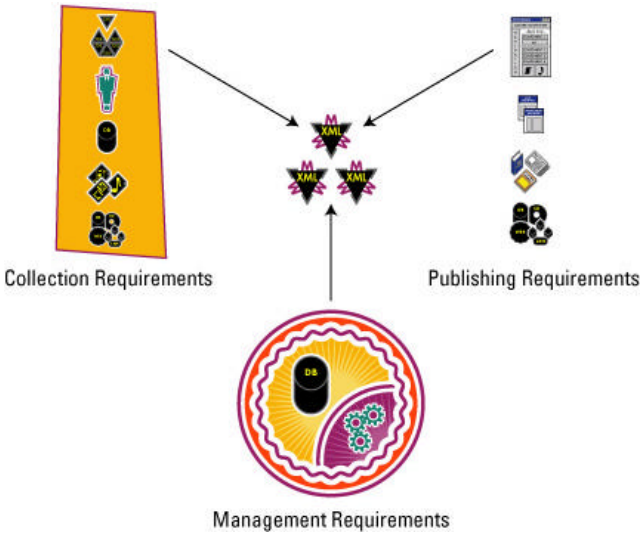


***Figure 3**: The more ways you use to define content components, the more sure you can be about them.*

By addressing the same target entity from a number of perspectives, you can be sure that you have accounted for all of its parts. Of course, the same notion works for any other part of the system. For example, I could have triangulated on publications by looking at them from the perspective of the components, the audiences, and the sources. Any combination will work.

Triangulation, or cross-correlation, as it is more properly called, is more than just an interesting exercise; it is the way to get at and solidify the requirements and design of your system.

Managing the details

Don't be surprised if you end up with hundreds of pages of data when you have fully fleshed out all of your design. For each component class (and you may have dozens), you might need three or four pages to detail its elements, element values, source information, target, audiences, the publishing templates where it shows up, and the personalizations that access it. For 25 types of components, that is already 100 pages!

The volume of information is increased by the fact that you will need to cut the same information many ways. When you are detailing a component class, for example, you should mention the audiences it serves, but when you are detailing an audience, you need to mention the component classes that are targeted to it. The combinations are numerous. To keep all of the information straight, I have employed a variety of word processing systems and spreadsheets, but nothing has worked as well as a database. In my current version, I have all of the design constraints I can think of nicely organized into tables. The relationships between the classes of constraints

(between components and audiences, for example) are specified in database relationships between tables. The database enables me to:

- Have one central place where all of my constraints are typed
- Account for the relationships between constraints as well as the constraints themselves
- Find and sort constraints
- Slice through a set of constraints and their relationships in any way I desire without needing to continually cut and paste
- Produce well-formatted reports

My approach may be overkill for you, especially at the beginning of your project, but keep in mind that you will eventually have all of the same needs that I listed. So, whatever system you use to type in and keep track of constraints, it needs to be robust.

Taking Stock of the Deliverables

You can use the following deliverables to form the basis of your requirements and logical design project:

- **The requirements plan of attack** charts how you will get to all of the people with whom you need to talk in your organization.
- **The requirements document** solidifies, and makes official, the requirements for the CMS project.
- *** The audience analysis** presents your selected set of audiences for comment and acceptance by the organization.
- **The design plan of attack** charts how you will move through the logical design process.
- **The collection, management, and publishing design documents** summarize the results of your extensive logical design process.
- **The localization plan** shows what you consider the most practical approach to localization for your organization to understand and approve.
- **The risk assessment and project plans** that you developed earlier can be updated and deepened considerably based on your requirements gathering and logical design processes.
- **An executive summary** can help you inform, and continue to receive, support from your organization.

The requirements plan of attack

If you prepare a little, your foray into the organization to collect requirements will be much easier. In addition, in the heat of collection, you will be able to refer back to your plan and not have to remember what to do next.

One way to approach this plan is to focus not on the requirements but on the people in the organization from whom you will collect them. My assumption here is that, given the questions I have laid out, you already know enough about the requirements you need to capture. On the other hand, I assume that you need to know more about how to collect them from your colleagues. To work this through, you can create a spreadsheet that lists your requirements contacts. Before you send out any questions, get together with each contact and define the following:

- **Which questions will you ask?** You might want to number your questions to make it easy to refer to them.

- **What kind of turnaround time** will you expect on answers?
- **What kind of review** of your synthesis will she require?
- **Who would be the right person to contact to escalate** any responses to questions that she is not comfortable answering to someone of more authority?
- **Who are the other people who should answer the same questions**? You can compare her answers to this question to the people you have targeted to be sure that you have not missed anyone, and that she is aware of the other people you have chosen for each question. It is better not to surprise her later.

With this task complete, you can create a second spreadsheet that focuses on requirements. For each requirement, you can enter the following information:

- The names of the respondents
- The overall turnaround times for responses
- The review process you will use for the requirement
- The arbitration process for conflicts in answers

The second spreadsheet is a plan of attack for gathering requirements.

Note

The length of this process is another reason I boil requirements down to so few questions. You could not practically do this sort of process with dozens of questions.

The requirements document

You need an official result of the requirements process to hang next to your project mandate. When the arbitration is finished on the last question, I suggest that you gather the final results into a document and get specific sign-offs from the following people:

- The respondents to the requirements questions
- Your sponsors or their assigned point persons
- Your project team

I believe that it is worth making this much fuss over the requirements document because it is the last of what you will expect the organization to provide to you for a while. As such, you want to make sure it gets full closure and that the signatories know that their chance for major input is now done. In addition, these requirements are your true marching orders. If the mandate is the goal of the war, the requirements are your orders. What is left, of course, is a battle plan.

Be sure to include in your requirements document clear references to the mandate. You want to be sure yourself first, and then communicate to others, that the requirements derive clearly from the mandate. For your own sake, as well as the effect it will have on others, be sure that this is actually true. Later, it may be too late to reconcile those nagging little inconsistencies between the overall goals and means you have chosen to achieve them.

As you might have guessed, my major organizing principle for all content management materials is collect, manage, and publish. To the extent that you can organize your requirements document with these three categories, it will clearly tie into the later deliverables as I describe them. However, you needn't go too far with this. The requirements document is the pivot point between the world of the organization and that of the CMS. The categories content, system, and publications that I presented earlier map very loosely to collect, manage, and publish, but are about half-way between the language of the organization and that of the CMS. They may be a bit clearer to the audience of the requirements document than collect, manage, and publish are.

The design plan of attack

There is a lot to do to fully develop and document a logical design. (It is for good reason that I devote so many chapters to it.) In planning for requirements, you could get away with a couple of spreadsheets. In planning for design, you had better trot out the best project management tools you have. Up until the end of requirements, the project is relatively small. Starting with design, it can begin to explode. It is the number of details and the number of relationships between details that cause such a large expansion of the project workload during logical design.

It behooves you to get organized early in the logical design phase and - stay that way. The result of not being organized is a longer duration for this phase of the project and a lot of missed details and relationships that may surface later in the project and haunt you.

The main things to account for in your design plan of attack are the following:

- **How you will collect and manage** the detailed information you will gather. As I mentioned earlier in this white paper if you can manage to get a database together, it will repay the effort.
- **Who will help gather** and process the information. Obviously, the same people who provided requirements will be of assistance. Be careful, however, not to count on people with a lot of other responsibility to do substantial work for you. If you don't already have some, this may be the time to get some staff of your own.
- **How you will time the data collection** and processing. As one example in many, it is just as valid to work backward from publications to components as it is to work forward from components to publications. In either case, how will you ensure that, in the end, components and publications are completely tied together? I mentioned the techniques of triangulation and iteration earlier to help you find ways to figure this out. In your plan, it is not enough to say that you will iterate, you must say how long each cycle will be and what exactly it will accomplish.
- **What you will deliver.** I'll provide a general framework for the design deliverables, but it is up to you to turn it into a solid plan.
- **How you will deal with overload.** Be prepared at the start for the amount of detail to overwhelm you. As you move forward, how will you reset expectations on required content and publications if you can't do them all? Don't wait until you have run out of time to confront this issue.

I prefer to use a project Gantt chart to show all of the milestones to be accomplished in the design process (shown in Figure 4).

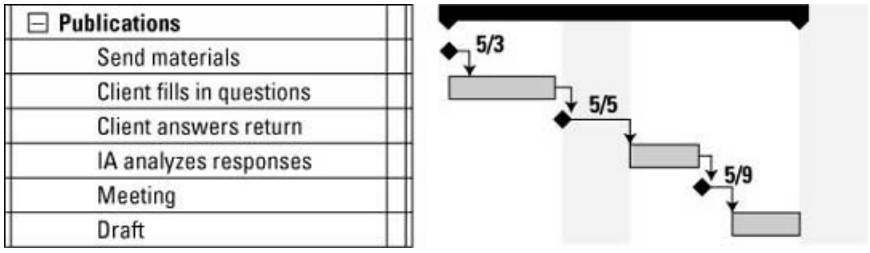


***Figure 4**: A Gantt chart can help you stay organized and on target during design.*

Based on what is most convenient for your team and organization, you can create one big logical design document, or a number of smaller ones that fit together as a set. As you might expect, I will organize the information to be presented in three parts: collection design, management design, and publishing design.

The collection design document

To logically design your collection system, you decide, and then document, how content will move from outside the CMS to the point where it is ready to be used in publications. Following my discussion of what occurs in a collection system, the collection design document will have to cover these topics:

- **Who are our authors** and how will they be tied into the CMS?
- **What are our acquisition sources** and how will they be tied into the CMS?
- **What components will be created** by which authors and sources and at what rates?
- **What conversion processes will we need** to transform the format and structure of the information we receive?
- **What staff will be needed** to collect content and what will their tasks and processes be?

You can use the information provided in the respective chapters that I reference in the preceding list to decide what you need to say about these topics. As far as how to construct the document, consider its audiences. First, the staff it mentions will want to read this document. It should therefore have well-marked sections that divide responsibilities by position name. Next, specification writers will want to read it to understand how to set up the collection systems. Thus, you should make sure that it divides the information by functional units (for example, authoring tools, conversion systems, acquisition procedures, and so on). Finally, it will be used as input to the system selection process, so it should call out any information that should be included in the selection process.

If you are starting to think that you need a CMS just to produce the various cuts on the design of your CMS, you are on the right track.

The management design document

To logically design your management system, you decide on, and then document, your approach toward content components. The design has the following basic kinds of information:

- **Information about components,** including which ones you will create and how they will be meta-tagged. I call this the content schema.
- **Information about how components will be cataloged and accessed,** including the hierarchies and indexes that the system will use to track and publish components.
- **Information about how the components will be managed in the CMS repository,** including how you will track them through their lifecycles and eventually retire them.

The content schema

A content schema divides content into components and components into elements. The schema specifies the following:

- **The set of components** to be managed.
- **The set of elements** for each component that includes both content and management elements.
- **The allowed values** for each element.
- **The subset of the elements that are shared** between a number of components to keep consistent naming. For example, you might have an element called "Illustration" that is used in any of the components that require a demonstration image.

- **The subset of elements that are universal;** that is, shared between all (or almost all) components that will be used for accessing and managing them. For example, you might decide that every component will have a "Status" element so that you can track its lifecycle stage.

To construct a content schema, you divide and organize your information until every part is accounted for, and you can confidently say that you have a workable system of content and management elements and the components of which they are a part.

To deliver the schema, you document the components, elements, and values. A spreadsheet or word processing document may suffice. If you use these methods, your problem will be how to cut the same information in different ways. For ease of use later in the process, you will want to deliver the information sorted by component as well as by element. This is cumbersome, but possible, in a spreadsheet application, but a real pain in a word processor. For that reason, I tend to use a database application where I can switch easily between component and element views of the information.

Just as components are the center of the content management process, so the content schema is the center of the logical design process. This means that it depends a lot on the other parts of the design. Therefore, don't expect the schema to come out quickly and stay the same. Rather, it will come out slowly and evolve throughout the logical design process. It will continue to evolve (though much more slowly) throughout the implementation process. Even after your system is running, you should expect the schema to change as you respond to the changing needs of your information marketplace. At that point, of course, it is maintenance, not design.

The content schema is the starting place for both the schema that you will need to implement your system and the metatorial guide that you will need to produce for system deployment.

Note

If you know that you will be using a relational database for your repository, then you can deliver the schema to the implementers of the system as a database design. If you will be using an XML system, then you can deliver it as an XML DTD or schema. If you do deliver using relational database or XML methods, then you will also need to create some sort of document to deliver notes, caveats, or values, and relationships that can't be well-represented in these tools.

Component access

A big part of the logical design of the CMS is to figure out exactly how to catalog components so that you can find them and automatically generate the navigation of your publications.

The feature that you cannot do without in this section of your management design document is the component hierarchy. The hierarchy is an outline that categorizes all of the components you will manage. By creating this hierarchy, you are sure that all of your components fit together in a single access system. It is equally important to show how this overall component hierarchy can map to the hierarchies of each of your publications. If your system must produce components in multiple languages, then you can show the relationship between the versions in this hierarchy.

You can also include the following sections in your management design document:

- **Indexing terms.** It is less important that you actually come up with an exhaustive set of terms than that you come up with the first list, and also decide how the master list of terms will be generated (when and by whom), as well as how to translate the master list into the indexes that you include with any publication that uses an index.
- **Cross-reference strategy.** Here the point is not at all to create any cross-references, but rather to decide on general policies for when they are needed, how they will be created, and how they will be allowed to be represented in each publication.

- **Sequencing strategy.** Again, you will not be in a position to actually create any component sequences, but rather to create the policy for how sequences will be represented.

Component administration

In addition to putting together a component schema, during logical design, you need to decide on, and document, the staff, processes, and tasks that components will have assigned while they are in the repository. These processes include the following:

- Periodic editorial reviews
- Periodic metatorial reviews
- Archiving procedures
- Versioning procedures
- Deletion procedures

The form this deliverable takes is up to you, but it should match the way you document staffing and workflow in the other parts of the logical design.

An audience analysis

In addition to the data you gather on audiences and how it plays into your other analyses, I think it is a good idea to produce a specific deliverable on audiences for the following reasons:

- **Perspective:** The conclusions you reached on audiences ought to be out in front of the project, giving the rest of the organization the same perspective it can give you. You want the members of your organization to think first about the people served by the CMS and only then about their own perspectives.
- **Purpose:** For many organizations, there is value simply in stating what the core audiences are. If there are disagreements and different audience sets in use throughout the organization, then your report may serve as the one point of officially sanctioned segments. Others can adopt it, or at least orient to it, as they work though their own audience analyses.
- **Use:** In addition to the list itself, others in your organization are bound to find the information you have collected about audiences at least interesting, if not incredibly useful. If you do a good job, then you will save the organization from the recurring expense of doing the same research over and over again.
- **Feedback:** Of course, the information flows both ways. By publishing and publicizing your audience analysis (under a more enticing name, please), you will elicit feedback from competing initiatives as well as from closet experts who know more than you could ever hope to about some aspect of some audience.

In most organizations, the work you do to analyze audiences will be ahead of the pack. Other analyses are likely to be either less broadly focused or not as thorough. Your analysis needs to be both broad and thorough to fully power a content management initiative. On the other hand, because your success depends so much on the audience analysis you have done, you need to protect it from frivolous attacks from competing schemes. If your analysis is well-known and supported, other schemes will have to prove merit, not just support, to challenge it.

In the end, what you want is not to be the one to define the organization's market segmentation, nor be constrained by audience definitions that are inconsistent with your other plans. You want the CMS initiative to be a player in the ongoing determination and refinement of your organization's constituencies. It would be less work for you if someone could simply hand you a good analysis already done. On the other hand, the CMS perspective can deepen and make more tangible the analyses that others have done.

The publication design documents

Publication design is somewhat more involved than collection or management design. Because each publication has its own interface (even books have a paper interface) and visual design qualities, there may be quite a bit to do to fully design even one publication, let alone the variety of publications that a CMS may be called upon to create.

> **Note**
>
> It's beyond my expertise to offer process or advice on the aspects of publication design that are outside of the content management arena (graphic design, branding, styling, look-and-feel, and the like). Instead, I'll focus on parts of publication design that are directly created by the CMS (for example, not the nature of the branding, but rather how branding elements created outside the CMS will be laid out on a publication page by the CMS).

In publication design, you define and prototype the pages of the publications the CMS will be called upon to produce. The goal of publication design is to create the most compelling publications possible that can still be efficiently produced from the CMS. Ideally, publication design should begin at the same time, or even before, you begin collection and management design. Collection and management design ensure that a viable system can be created behind the publications, and publication design ensures that the system produces compelling output.

The deliverables from the publication design process can be a large number of files rather than the few files that result from a collection or management design process. These deliverables include the following:

- **A publication strategy document**, which specifies how you will deal with your publications in general
- **Page design files**, which define, at increasing levels of detail, how the pages will look and be built
- **Personalization strategy documents** that show how you intend to target and deliver personalized content
- **A publication administration document** that details who will be needed to create publication on a start-up and run basis, what tasks they will have to accomplish, and how frequently they will have to accomplish them

Publication strategy

Although each publication is its own world with its own constraints and design, the set of publications that you will create need to hang together as well. To figure out how individual publications fit into the whole, you can create these deliverables:

- **A component-to-publication analysis:** This document tracks each component from its neutral state in the CMS repository to its display in various publications. It ensures that you know how each component will be used in the final publications and that there are no conflicts between them. The core of this document is a large table (word processing, spreadsheet, or, preferably, a database report) that summarizes all of the relationships between publications, pages, and components.
- **A template design document:** This document describes the overall approach you need to take to your publication templates. It includes a discussion of any sharing or commonality there will be between publications, as well as sections for the templates for each separate publication. In the publication-specific sections, the names of the page templates to be used are listed, as are the static text, component, and navigation templates that will be included in each page template. Finally, for each template mentioned in the document, there is some

definition of the logic that the template will have to include in order to access and assemble its content properly.

Page design

Page design concerns the look, layout, and building of publication pages. The deliverables for this process that have the most bearing on a CMS are as follows:

- **Page wire frames**, which are schematic drawings of pages with boxes where the content will go. The wire frames name the various areas of the page and show their overall layout and size. Because of their simplicity and lack of detail, it is often a good idea to do a set of wire frames as a first step in your publication design, to get an overall view of the pages you intend to create (as shown in Figure 5).
- **Page mock-ups**, which are illustrations of completed pages that are produced so that people can get an early view of what a final page might look like. Sometimes called comps, these illustrations enable you to see an average example of the pages in your publication without going though the effort of actually producing them. Be certain, however, that your page mock-ups adequately reflect the diversity of content that may appear on a page of a single type.
- **Page prototypes** are page mock-ups done in the actual publication medium. For example, an HTML prototype would look the same as the corresponding page mock-up. The page mock-up, however, is only an image, whereas the HTML prototype will actually run in a browser, just as the final site will. You can use page prototypes to prove that your designs will work and, also, to test the variability of content that may find its way onto a page. For example, if you have decided that you can have from one to ten components on a page of a particular type, then you can quickly add components to, and subtract components from, the prototype page to make sure the final page will always look all right. Prototype pages created as part of the logical design process do not generally include any logic or other CMS programming code. They are simply there to make sure that pages can be created in the target environment as desired.

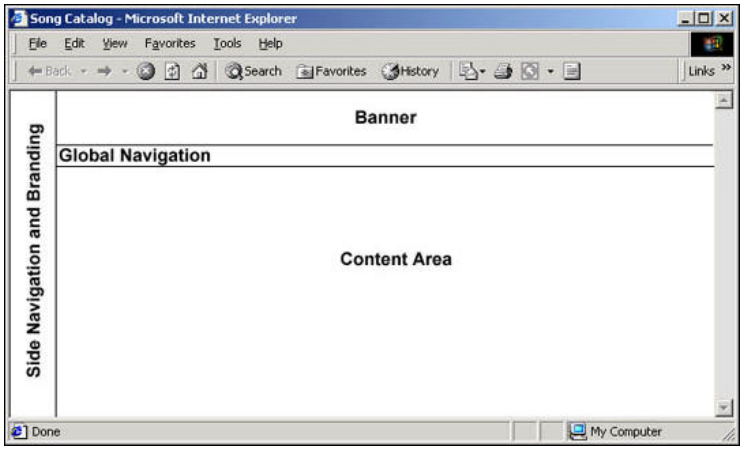


***Figure 5**: This is a very simple page wire-frame diagram that shows the locations and sizes of the areas of a Web page in the context of a browser.*

Personalization strategy

Even if you will have only a couple of audiences, for starters, and a static Web site as your only publication, it is worth your while to study and document an approach to personalization. This approach can include the following deliverables:

- **A messaging strategy:** This document combines audiences, publications, and components together at a high level. It defines what each audience is supposed to get from each publication and which components and publication design features will be used to communicate with them. It is a high-level precursor to personalization analysis. Particularly for publications targeting external audiences, it provides a marketing context within which to do personalization. For internal publications, your messaging strategy clearly identifies which types of information are appropriate for each audience.
- **A personalization strategy:** This document goes into detail about exactly how you intend to do personalization in each publication. It specifies a personalization rule for each combination of publication, page, audience, and component. The core of this document is the set of personalization rules. The rules state how to fetch components from the CMS repository, based on what you know about the user, plus the values of the component's management elements. The rules also say which elements to display and how to lay out the components on the publication page.

Publication administration

If you have more than one publication depending on the same base of content, then it is important that you work through all of the scheduling and dependencies to ensure that each publication can get the content and staff attention that it needs. In addition, as part of the logical design of the CMS, you will need to decide on, and document, the staffing and workflows you might need. Following are the documents you will produce when working on this aspect of your CMS:

- **A master publication schedule** shows the proposed release dates of each publication, as well as the components on which the publication relies. Using this schedule, you can make sure that the component collection and management workflows you create conclude at the right times for each publication to be released, and that two publications do not put conflicting constraints on component completion schedules.
- **A publication staffing and workflow plan** the jobs, tasks, and workflows you expect the publication process to require.

A localization plan

Content and publication localization is difficult. It can add orders of magnitude to the budgets of systems that do not handle it well. Even for the most efficient system, localization can expand schedules and budgets enormously. Expectations for localization tend to be unrealistic, either way too high or way too low. If you plan to localize, it is crucial that you define and set reasonable expectations for what will occur and when.

Logical analysis is the right time to create and publish a localization plan. It is late enough in the project that you will have good information to base it on and early enough that you can debate and then finalize a plan in time to implement it.

The core of your localization plan is the set of localities that you plan to support and how you intend to support them. Because localization cuts across all of the parts of logical design and implementation, be sure you cover each locality from all significant angles, as follows:

- **Audiences:** Will you create audience types for each locality or will you create local segments within each audience type?
- **Publications:** How will navigation differ from locality to locality? Will you replicate templates or will you branch within them? What modifications are needed to your personalization approach to have it work across localities?

- **Components:** How will translated content be stored? Will you need any extra component elements to account for locality differences?
- **Authors:** Do local authors constitute a different author group or can they be grouped with other authors?
- **Acquisition sources:** What source material originates outside the core region? How will that information be processed and merged with content from the core region?
- **Workflow and staffing:** What are the specific tasks involved in localization? How long will they take and how many and what kinds of people will need to be involved?
- **Access structures:** Will structures be simply translated or will they be localized as well?
- **Deployment:** How is locally originated content distributed throughout the system? How will local and global content be merged and deployed to the appropriate servers throughout the world?
- **Support:** What support for translation and localization do you expect from the CMS that you choose?

A fair accounting on all of the preceding issues ought to give you the information you need to gauge the magnitude of the localization effort before you. Be sure that this analysis makes it out of the depths of logical design and into the hands of your sponsors or other decision makers so that they can either fund localization at the level it deserves or scale back the organization's expectation of how much localization will occur.

A risk assessment plan

As you proceed through logical design, you will make any number of assumptions that might turn out to be untrue. If a bad assumption could cost you significant time or money, then it is a risk. In addition, you are bound to want to suggest strategies that might not work. Instead of suggesting them anyway, or worse, not suggesting anything that you think might fail, suggest it, and assign it a risk probability. If the risk is too great, then you can drop the idea. If, as is often the case, the risk is close to the reward, or can be mitigated with a little extra effort, then it might be worth a try.

Here is a starter list of risks, from requirements and logical design, to get you thinking:

- **System requirements:** Do requirements conflict? Were you unable to get them down to a comfortable level?
- **Audiences:** Was there agreement? Are there too many audiences? Too few? Will you be able to get the data on each?
- **Publications:** Are there too many? Are they too complex? Will you be able to make all the integrations required?
- **Content components:** Are there too many classes? Are there more classes to come that you have not yet categorized? Will you be able to gather the metadata you propose to include with each component?
- **Authors:** Do you trust the information you have gathered? Are there too many different types of authors? Are they skilled enough to do their jobs? Are they close enough to you to follow the rules you create?
- **Acquisition sources:** Are there sources out there you could not analyze? Will some take too much effort to convert? Are you dubious about the quality or consistency of some? Are there too many?

- **Access structures:** Can you muster the effort to create the ones you suggested? Are they adequate to make content accessible? Will people really be able to assign components to the right categories? Can publication navigation really be built from the access structures you plan to put in the CMS repository?
- **Workflow and staffing:** Will the system take more effort than you can muster? Are your task time estimates accurate enough to make predictions from them? Can you find skilled people or train them? Are there bottlenecks in your workflows?

A revised project plan

At the end of requirements gathering and logical design, you know everything you need to know about what the CMS needs to accomplish. Although this is not all you need to know to lay out a final project plan, it is enough for a next hard look at the plan. You can choose to revise your project plan either after requirements and logical design, or just once after logical design.

After logical design, you should be in a position to revise that plan as follows, based on what you learned:

- **Staffing:** You should have a very good idea of the people and tasks you will need to start and run your CMS. This new information should help you infuse reality into your plan.
- **Quantities:** You now know exactly how much content you will have to process and how many publication pages per month you will have to produce. This ought to help you refine the schedules and budgets for these activities.
- **Level of development effort:** You should have a good idea, from both the functionality components you designed and the system requirements you gathered, of how much software development will be involved in your CMS project. You should be able to do some basic calculations at this point on the amount of time, staff, and budget you will need for programming and other technical services. Of course, it will not be until you create implementation specifications for all of the software development tasks that you will know precisely what it will take, but at this point, you should have a good first approximation.
- **Level of infrastructure:** From both the system requirements you gathered and the logical design you did, you should now be able to estimate the kinds and amounts of hardware and software you will need. Of course, until you select a CMS product and complete your implementation specifications, you will not know for sure, but there ought to be enough information available at this point to make reasonable guesses.

At this stage of the project, you should be able to revise a project plan such that it is clearly doable within the time frames and budgets that you have. As likely as not, when you first revise your project plan, you will find that you now exceed even your most liberal time and budget estimates. That is a sign that you have done a complete analysis. If this happens, you can either ask for more time and money or you can cut back.

If you need to cut back, the appropriate thing to do is to start at the mandate. What can you cut from your mandate that, in turn, negates certain requirements, which, in turn, negate components and publications? If you start, as most will, cutting components and publications first, then you will have done the following:

- Ruined the coherence and unity of the mandate-requirements-logical-design relationship that you worked hard to create.
- Lost a great opportunity to argue for more resources based on what the organization does not get to have for less. It is easy for the organization to let you cut content and publications and still hold you to the original goals (which is, after all, what they really care about). It is difficult for them to let go of the goals they themselves crafted.

Inevitably, you will have to cut back some of what you designed. You can be upset about the cutbacks, or you can take it as a sign that you have been smart and have designed ahead of implementation. It is good to make your current design bigger than what you can accomplish in your next implementation. A bigger design helps you be sure that the implementation is meeting current needs, and preparing you for the future.

Tip

Don't look at cutbacks as a defeat. They very well might be a good thing for both the project and the organization. If you cut the parts of the system that were the biggest stretch, you will be buying more time to really understand them before you actually have to do them. History is on the side of content management. Eventually, you will need to do all that you have designed and more. You needn't be in a rush to do it all at once, especially if you are really not quite sure how you could do it even if you had all the time in the world.

An executive summary

Sponsors, and anyone outside the project team, are unlikely to want to wade through the mountains of information you generate from the logical design. For these folks, it is best to create an executive summary that is fewer than ten pages long. Here is a list of the sections you might consider including:

- A reiteration of the mandate and major goals of the project
- A summary of the requirements you gathered
- An overview of authors and sources
- A shortened, annotated version of the component hierarchy
- The messaging and personalization strategy
- A few choice page mock-ups to show what the final publications will look like
- An overview of the major staff positions and workflows you worked out

Note

If you don't know what some of the items in the preceding list are, read on. They are all covered in the chapters that follow.

In addition, you might consider including the updated project plan (or some shortened version of it) as an appendix to the report.

Taking Stock of Your Staffing Needs

Table 15-1 Staffing Needs

Task	Leader	Participants	Notes
Requirements plan of attack	Content Manager or Business Analyst	All other analysts as needed.	Content Manager will at least want to supervise this activity.

Requirements document	Business Analyst	All other analysts as needed.	Content Manager may want to participate too.
Design plan of attack.	Content Analyst or Business Analyst	All other analysts as needed.	Content Manager will probably supervise this activity.
Executive summary	Content Manager or Business Analyst	All other analysts as needed.	Content Manager might be best suited to create and present this document.
Audience analysis	Business Analyst	Publications analyst, outside marketing staff, and all other analysts as needed.	The Business Analyst is in the best position to lead this effort, but will need to rely heavily on the other people in the organization who specialize in marketing or in serving a particular audience.
Collection design document	Content Analyst and Software Analyst	There should be strong input from the conversion analyst and input from all other analysts as needed. Leader will bring in content contributors and others from the organization as required.	If practical, the two analysts should collaborate, because it touches each of their disciplines separately. If this is impractical, then one should own the document and the other contribute a lot to it.
Management design document	Content Analyst and Deployment Analyst	All other analysts as needed.	If practical, the two analysts should collaborate, because it touches each of their disciplines separately. If this is impractical, then one should own the document and the other contribute a lot to it.
Publication strategy documents	Publications Analyst	There should be strong input from the Content Analyst and other analysts as needed.	The Software Analyst might need to participate in the templating section.

Page design	Publications Analyst	Content Analyst, Publication Designer, page Developer, and UI Specialist	The Publications Analyst should direct the effort, but the other staff members are the ones to create the deliverables. The Content Analyst should consult heavily on the parts of the analysis that concern the content component. The Publication Analyst or Publication Designer will need to bring others in for specific input.
Personalization strategy	Publication Analyst	Content Analyst and Software Analyst	The Publication Analyst should lead the effort, because the personalizations affect the structure of the publications. However, the Content Analyst will need to consult heavily on the access and retrieval of the content, and the Software Analyst might have to consult on the personalization rules and their implementation. Input from marketing groups also might be needed for the messaging analysis.
Publication Administration	Publication Analyst	Deployment Analyst and other analysts as needed	The Deployment Analyst can aid in understanding the role of infrastructure staff in the publishing workflows and tasks.
Revised project plan	Content Manager	All other analysts as well as a Project Manager as needed.	The analysts might be responsible for each part of the plan, but the Content Manager is in the best position to own the plan, make the tough compromises, and get continuing support for the plan.

Summary

Requirements and logical design are the beginning of the meat of a CMS project. My version of requirements has you ask the organization for the most general, but the most important, content, publication, and infrastructure needs.

My version of logical design has you work from these general requirements to a very specific design for what you want the system to accomplish:

- In doing requirements gathering, use common sense and organizational savvy.
- In doing logical design, be sure to iterate a number of times through the process and find a way to manage the tremendous amount of detail you will amass.

- Prepare a plan of attack for requirements and logical design before starting out on them.
- Prepare analysis documents for collection, management, and publishing (or along any other outline that covers all of the logical design bases).
- Give the organization's leaders an audience analysis and a localization plan that they can debate and, ultimately, accept.
- Give the organization's leaders an executive summary that they can easily read and thereby get a feel for what you are doing.
- Finally, revise your project plan and risk assessment at the end of logical design so that it reflects your most current thinking.