Keys to a Successful Document Assembly Project

By Marc Lauritsen and Alan Soudakoff

(A version of this article appeared as "Unlocking the Power of Document Assembly"
in *Law Office Computing*, June/July 1999, p. 70-77)

Contents

Introduction

Document assembly systems are today's power tools for preparing customized legal documents, and they are showing up in more and more firms and departments. Such systems offer some extraordinary benefits. High quality written work can be produced in a fraction of the time it takes with manual methods, skyrocketing the effectiveness of legal professionals and their responsiveness to clients. But realizing these benefits depends upon well-managed planning, development, maintenance, and support. Issues of people and process often dwarf the technical dimensions of this work.

This article is about the broad managerial challenges involved in designing and building document assembly systems. This is a vast subject, and we're just touching on some important highlights. Rather than presenting a grand methodology, we've gathered some issues to think

about. Our goal is to provide a framework of analysis that illustrates common issues and suggests useful approaches.

We will pay particular attention to major document automation projects—those that involve complex and interacting documents, for use in *group* practices, implemented by a *team*. Efforts involving simple documents designed for use by a single practitioner can benefit from some of the concepts reviewed here, but our focus is on the considerably greater challenges of projects that involve one or more of the factors just listed.

We set forth below five basic success factors and then 12 stages of a typical document assembly project. It's difficult to talk about many of these ideas in the abstract, and so we will use concrete examples as much as possible. Since HotDocs (from LexisNexis – see www.hotdocs.com) is the most widely used legal document automation software these days—and a product with which we've had extensive experience—most examples will involve it. But if you look beyond the specific vocabulary and technical details, you will see principles that apply to all of the software alternatives. (We consider DealBuilder, www.business-integrity.com, and GhostFill, www.ghostfill.com, to be the most significant alternatives at present.)

Five ingredients of a successful project

Five basic ingredients are essential to any law office automation initiative: an appropriate *concept*, the right mixture of *people*, adequate *tools*, relevant *knowledge*, and an effective *process*. Let's look at these in the context of a document assembly project.

1. Getting clear on the concept

The beginning of success is being reasonably sure that the concept being implemented is sound. Is what has been proposed doable given reasonably available staff, time, tools, and other resources? Do you have a fair shot at achieving a quality result at a reasonable cost and in a reasonable period of time?

Keep it simple. Managing all of the aspects and stages of a document assembly project can be like doing a three dimensional crossword puzzle. The complexity can easily get out of control. This complexity can be minimized to some degree by keeping the underlying project as simple as possible. Bear in mind an 80/20 rule of thumb: automating about 80% of the potentially automatable work will provide the best return on your effort.

Know your users. You should decide up-front who will be the main users of your system. A system designed for attorneys with deep knowledge of underlying substantive material will look and behave very differently from a system designed for summer interns.

Do your due diligence. Be sure that no adequate off-the-shelf product already exists. There are many commercially available systems for documents such as tax forms, LLC formation, probate

(see e.g. www.estateworks.com), estate planning (see e.g. www.wealthcounsel.com), immigration documents (see e.g. www.lexis.com/bookstore/catalog), and court filings in certain states. You don't want to reinvent the wheel unless you are confident you can do a better job and have the time and money to do so.

Identify the anticipated benefits. You want to be sure the project is well justified, that its benefits are clear, and that some identifiable constituency's bottom line will be directly improved by the application. (Management consulting jargon: be "aligned with the business strategy.") You may not need to do a formal return-on-investment analysis. But insist that people articulate the organizational value, the business rationale, of the project. Why are we building this system? What problems or opportunities does it address? Who will benefit? How? What do we expect to achieve? Faster turnaround? More competitive prices? Better quality? Clarifying expected benefits will not only confirm the business case for the project, but guide the effort as design crossroads are encountered.

Address the sensitive billing and compensation issues. Will people be motivated to help with and use the system? Has someone thought through how the office will recover the investment if the system results in less time being billed for similar work? Knowledge leveraging technologies like document assembly will yield greatest benefit to firms willing to innovate in how clients are billed and colleagues are rewarded.

Manage expectations. Document your decisions on the scope of the initial project and circulate it to your project team so that expectations are properly managed. Be realistic about how much—or little—work processes will change when the system is in place.

2. Assembling a cast of characters

Legal document automation projects are inherently multidisciplinary. Doing them well involves drawing on several varieties of legal, computing, and managerial skills. You need substantive legal knowledge, technical know-how, and facility in handling people and processes. In some rare cases these skills can be found in one person. More typically they are assembled from different folks. A document assembly project team could include the following:

a. **Legal experts.** These are the attorneys or other legal professionals who have special knowledge of the subject matter and are the primary authors of the model legal documents.

b. **Project manager.** This person keeps the project on track, runs the meetings, interfaces with any consultants, and resolves disagreements.

c. **Template developer.** This person develops the document templates and any related programming.

d. **End-users.** These are the ultimate users of the system; they may or may not include the legal experts.

e. **Trainer.** This person trains users on the operation of the system. He or she may also prepare or edit the documentation.

f. **IT professionals.** These people install the software, assign rights to shared files, and integrate the system with databases or other third-party applications.

Sidebar 2 shows how each of these players might be involved in the 12 stages of a document assembly project. Here are suggestions on assembling your team:

Appoint a design leader. Give one person—typically a mid-level attorney and often the project manager—primary responsibility for settling questions of design: what words should go where under what circumstances in the documents, how dialogs should be laid out, what the help screens should say, what features should be used in what ways. This person of course is not a dictator; he or she will elicit and synthesize views of the team and others. He or she may not be the ultimate decision maker, and will likely need to run some decisions by colleagues or higher authorities. But *someone* should be charged with speaking authoritatively on design questions so that debate can end and work can proceed.

Involve users throughout the process. Be sure that at least some of the ultimate users of the system play an active role in the development process. Not only will they bring unique insights into the work sought to be automated, but their involvement will help ensure acceptance and eventual ownership. "Participatory prototyping" is a slogan that captures the spirit of iterative product development with active user involvement.

Bring in experienced help. If your team lacks experience in doing projects of this kind, seek knowledgeable assistance elsewhere. Bring in someone from another office or practice group who has gone through the process. Consider engaging help from the vendor of your document assembly software or an independent service provider. If you can find developers with legal training, that will reduce some of the cross-disciplinary miscues common between lawyers and technologists. There are quite a few independent consultants these days who fit this description. Many law organizations actually outsource the entire development function, and sometimes even ongoing maintenance and support.

3. Tooling up

The questions arise: With what software will we tackle this project? Can and should we use our existing word processing software, or acquire some specialized document assembly package? What packages are out there? How do they and their vendors differ? Which offers the best fit for our needs?

These questions may be answered by organizational precedent, consultant recommendation, or independent research. They may be answered at the very beginning of the project, or postponed until a fuller vision of the intended application has been developed. The latter way, the best application for a particular project can be chosen. Delaying the decision for too long, however, can hamper the development of a prototype system.

We've previously written about available tools. (See our now considerably dated Shopper's Guide to Legal Document Assembly, *Law Office Computing*, October/November 1997. It and lots of other product reviews are available at www.lawofficecomputing.com. For more up-to-date information, see e.g. http://www.lstech.org/workgroups/doc_assembly, which focused on web-based tools.) This article assumes you've already got that choice behind you, and focuses on the planning and implementation processes involved in a particular project.

Choosing a document assembly platform is not a til-death-do-us-part commitment. You will likely want or need to change every few years. Be aware though that it is often difficult to convert document assembly systems from one product platform to another. Even market leader LexisNexis does not have a simple migration path or automatic process for converting systems between CAPS, its legacy product,and HotDocs.

4. Getting in the know

Successful document assembly projects require the deployment of considerable knowledge—about the law, about software and its design, about managing people and processes.

A little knowledge can go a long way. But usually not far enough. Even relatively simple tools like HotDocs harbor surprising complexity, with plenty of traps for the unwary. Simple tips like frequently backing up, and testing parts of a template at a time to locate a missing "END IF" can save hours of frustration. Knowing what strange behavior is a known bug and what is a "feature" also can ease anxiety. There is a great deal of unwritten lore about its odd limitations, "gotchas," and obscure tips.

Mine your firm's substantive knowledge. Interview all the relevant experts in your department or firm. Assemble all relevant document models. Prepare decision trees and ask your resident experts to comment on them. Do some background reading in the vast and emerging discipline of legal knowledge management.

Read the manual. HotDocs has some of the best documentation in the business, although it is no longer supplied in hardcopy format. Resist the urge to plunge ahead without reading it. But even so, this official documentation is limited, and runs out even on some basic questions quickly encountered by new users. And written manuals are particularly ineffective at conveying the spectrum of choices developers face at various points, and the cascading implications of those choices. So you have to cultivate other channels of information, like the following.

Network. A lot of people have now done document assembly projects (successfully and not). Get in touch with them. HotDocs enthusiasts have organized user groups in several parts of the U.S. and abroad. If you're close to one, participate. If not, consider starting one. Attend one of the national HotDocs conferences held every few years. (The 1999 training and development conference attracted over 200 participants from 38 states, although Lexis has not organized such a conference in recent years.) Dig out written wisdom from fellow travelers, like Eidelman and Maeso, 27 Tips for a Successful Document Assembly Project, June/July 1996 *Law Office Computing*, p. 54.

Join a list serve. HotDocs and GhostFill maintain "listservs" (e-mail-based discussion groups) about their products, through which flows a great deal of practical information. New users often post seemingly basic questions, and many get answered quickly by helpful strangers. We find it useful to maintain a folder of particularly informative messages. Archives of the HotDocs list are maintained at http://support.hotdocs.com/mailman/listinfo/hotdocs-l.

Use the Web. Vendor websites are great sources of information. Extensive technical notes are available on many topics, as is a searchable database of known problems and fixes. If you search the Web for "legal document assembly" and are willing to sift through the results, you will find some decent literature from vendors, consultants, and academics. For HotDocs in particular, the "Computation Archive" maintained by Jim Garrison (http://www.legalcs.com/hotdocs/index.htm) is a wonderful resource, as is Lexis's own HotDocs "Developer Corner" (http://support.hotdocs.com/developer) .

5. Managing the process

These projects can involve an exquisite choreography of technical, managerial, and legal-professional tasks. Here are some suggestions on managing such tasks, whether you are an in-house developer or outside consultant.

Plan, plan, plan. Running a successful document assembly project involves a lot of common sense ideas applicable to other technologies and good management generally. Decide who will do what to implement your design. Set a schedule—do a rough realistic estimate of the hours, then double it! Set up review meetings for your team. Try to keep discrete projects to a few months at the most so users can see some concrete results early.

Anticipate issues and problems. A document automation project involves hundreds of implicit and explicit design decisions. These can be dealt with ad hoc, as they arise, or anticipatorily. We recommend that you anticipate as many as possible, and come to explicit decisions that will guide action as similar issues present themselves.

Be prepared to compromise. Good managers recognize that projects are full of balances, trade-offs, and calculated gambles. You will at least implicitly be balancing structure and flexibility in the process, balancing elegance and expedience, and trading off thoroughness for

reasonableness in time and costs. For instance, sometimes the need to move quickly outweighs the value of having finalized document models before you start.

Overdo it on communication. For those other than the project leader and template developer, these projects are generally a matter of peripheral attention. Substantive experts are busy practicing and developing business; users are consumed with current ways of doing things; office IT personnel are putting out fires elsewhere. For a project to remain coherent and sustain momentum, take it upon yourself to communicate to the point of apparent excess. Send e-mail to remind folks of meetings and tasks; circulate updates of project plans; hold frequent show-and-tell sessions along the way.

Get management on board. Having senior personnel on board early, and with a stake in the outcome, can provide substantial motivation. Invite them to major meetings. Encourage them to go public with their support.

Tag, you're it

Okay, let's say a senior lawyer in your firm or department has asked you[1] to be the project leader for a system to produce organizational documents for clients who want to form new corporations. You've gathered some sample documents and maybe read about some document assembly software. Now what do you do? There's no need to panic. You're just building a delicate little "soft machine" that will be used by some very smart, very busy people in the midst of urgent and complex work on behalf of demanding clients. That's all.

Well, maybe you should panic a little. Better yet, become acquainted with the five success factors for an effective document assembly projects, outlined above, and understand the stages of a typical document assembly project we set forth below.

The life cycle of a document assembly project: a 12-step program

A document assembly project usually involves the following dozen phases, or stages. The stages do not necessarily occur in this order or in neatly consecutive fashion. In fact, they typically overlap and interpenetrate.

1. *Defining the Project*
2. *Designing the System*
3. *Planning the Implementation*

1 "You" – the project leader – could be an attorney, a paralegal, a secretary, or an IT professional and can assume one or more of the other roles we identified earlier. As lawyers with long experience in document assembly and other aspects of law office technology, we've seen projects successfully managed by people with a wide variety of backgrounds.

4. *Settling the Document Models*
5. *Selecting a Software Platform*
6. *Building the Technical Environment*
7. *Developing the Templates*
8. *Testing and Refining the System*
9. *Writing Documentation*
10. *Rolling out the System and Training Users*
11. *Maintaining the System and Supporting Users*
12. *Bringing out Version 2*

Stage 1: Defining the Project

The considerations in "Getting clear on the concept" above are applicable here. For the incorporation project we described, following the 80/20 rule means limiting your initial project to perhaps one state where the majority of your clients choose to incorporate. You can build on your success and add additional states later. It also may mean limiting the organizational documents you will generate—perhaps start with the articles of incorporation and a cover letter to the appropriate Secretary of State. You can always add the bylaws and other documents later. If certain types of corporations require very specialized paragraphs or clauses, and they are only a minority of your clients, consider leaving these companies out of your initial system. For example, your initial system may not need to include "S" corporations or non-profits.

As stated earlier, deciding who will be the main users of your incorporation system is important. For example, a system for first year associates may need explanatory text and help screens on when and why to choose certain clauses. This is particularly true if one of the goals of the system is to educate new attorneys. A system geared toward senior attorneys, on the other hand, needs to provide less substantive guidance but may need to be particularly easy to use given the limited time and computer experience typical of their generation.

This is also the time to investigate what off-the-shelf systems are already available to draft incorporation documents in your state. This is an important step as there are likely to be products available that could save you time and money. For the purposes of this article, however, we assume that you have determined that there is no suitable product currently available.

Finally, you should decide if you will access information from an external database—such as a database with the names and addresses of your clients. This will help reduce errors in your system and speed up the document generation process, but will complicate the project.

Stage 2: Designing the System

There are many design issues when engineering a document assembly system. Some "conceptual" aspects of design are important to frame early in the process. We touch on some of those here; other more technical design issues are mentioned below in Stage 7.

How many discrete *documents* will your system include? For example, a letter to the Secretary of State may be re-used a number of times in your system, so it is best to make it a discrete template. It can then be designed once and then "inserted" as needed in a variety of other templates.

What general *features* will your system have? Do you want to provide a hierarchical look to the system so users can see the big picture as well as drill down to the details? Will you be doing any calculations? Are there some user errors that you want to anticipate and check for?

What *variables* will you use? Once the legal experts provide you with samples of the documents your system will produce, you can begin to prepare a list of the key variables. In our example system, this list would include the names of the incorporators, the location of the corporation, the type of corporation, the state of incorporation and the classes of stock. Determine the "data type" (e.g. text, number, true/false, date) of each variable. Decide whether any of the variables should have a default value—for example, should the state of incorporation be preset to Delaware?

What *dialogs* will you use? HotDocs allows you to group variables into screens that are called dialogs. Prepare an outline of these dialogs and their sequence. Does it make sense to ask for the address of each incorporator together with the name of the incorporator? Should this be done up-front or at the end of the series of dialogs?

What *help features* do you want to provide? For example, do you want to make available to users any provisions of the state corporation code? Are there some key tax issues that users of the system should be reminded of? Will this help be internal to the system or external, such as a Lotus Notes database or intranet web page? Again, go overboard in simplicity at first. There is always room for growth.

Don't stop thinking about tomorrow. If your current system only covers corporations but the goal is ultimately to include limited liability companies as well, are there any elements that should be built into your initial system so that it can eventually accommodate both types? If you need to ask for the number of incorporators, make it a number variable now (instead of a text variable) if there is any chance that you may want to use this number in a numerical calculation at a future time.

Stage 3: Planning the Implementation

The considerations set forth in "Managing the process" above are relevant here. Set a schedule for when the remaining stages of the project will take place. Who is going to be responsible for

making sure that the model incorporation documents have been approved by all relevant partners? When will automation of the forms begin and who will do that work? When can users expect an initial system to be rolled-out? Who will be assigned the unglamorous task of writing the technical documentation?

We like to maintain a written action plan that memorializes key assumptions and decisions about the goals and methods of a project together with an accompanying timetable. Circulating revised versions of this plan among the project team keeps everyone informed and invites feedback on what needs to be changed.

Stage 4: Settling Document Models

It is important to start with relatively stable documents before the actual template building begins. Once templates are developed in HotDocs, it is of course possible to change their language, but if there are wholesale changes to passages that have been heavily automated, you may be better off starting over.

To avoid this, circulate draft incorporation documents to all relevant attorneys and have each mark them up with preferred language. Given attorneys' propensity to prefer their own language, don't start automating until you have achieved reasonable consensus. You may have to lock all attorneys in a conference room for a few hours to do this. This is also a good time to seek consensus on style issues—settle on fonts, paragraph numbering styles, headings, and signature pages as early as possible to avoid tedious changes later on.

We have found that it's very valuable to have someone with content knowledge mark up documents with explicit instructions about how they vary from transaction to transaction. (Programmers sometimes refer to this as "pseudocode".) For example, bracketing all variables and inserting simple IF/THEN statements to show what and how paragraphs and clauses are conditional is quite helpful both to reviewing attorneys and the template builders. Mark-up can take many forms, from handwritten notes in the margins to careful electronic coding with a controlled syntax and vocabulary. Decide how much formality you need and evolve your mark-up conventions as you gain experience.

Stage 5: Selecting the Software Platform

Authoring software is often chosen in advance, as indicated earlier, particularly where there is a strong preference among the development team or where the firm or law department has other document assembly systems in place. If you have the luxury of choice, take a look at least two products. Are the incorporation documents you need to design official, fixed-form governmental forms? If so, you will want to chose a product that can automate such forms. See "Tooling Up" above for further discussion. You will also need to consider whether your application calls for one of the new Web-based document assembly platforms (e.g., HotDocs Online, GhostFill Server, or DealBuilder), and what if any database connectivity you may want to implement.

Stage 6: Building the Technical Environment

There are many technical issues to consider apart from those inherent in automating specific documents. For example, where will your templates and answer files be located? If you will have multiple users of your incorporation templates, putting them on a network directory eases distribution of updates. If several people sometimes need to work on the same incorporation documents for a single client, it makes sense to put answer files on a network directory as well.

There are lots of questions of just how a document assembly package gets installed on a network, and how its various setup options should be configured. Integration with other applications such as document management systems and databases is often very tricky. Don't underestimate the time and headaches for this aspect of the project. It is best to consult with the vendor and see what integration has already been accomplished. For example, HotDocs can read information from several specific case management systems and from databases generally with the HotDocs Database Connection, now included with HotDocs 2005.

You need to decide how many user licenses to buy. It rarely makes sense to purchase licenses for an entire firm or law department, or to put document assembly software on someone's desktop when no forms relevant to their practice have been deployed. For your incorporation system, start with enough licenses for the initial group of professionals who will use the system.

Stage 7: Developing the Templates

Building the document templates opens up another level of design issues. We highlight a few.

Plan your work. Do a mental run-through of the entire development process before starting. Visualize yourself working at each stage, paying particular attention to how the order in which you do things promotes or impedes efficiency. For instance, it's a lot easier to insert variables in a bunch of paragraphs *before* you break them into clauses than after. There is considerable value in just stepping back from the immediacy of your tasks and *thinking* about methodology.

Use a modular design. Try to re-use elements of your design as much as possible. For example, if you need to test several times if there is more than one incorporator, write a piece of code—a "computation" in HotDocs—to do this once and then reference it wherever necessary. This not only will make the templates less cluttered but will greatly simplify maintenance. It is a lot easier to modify a computation once than to find the dozen places the same code is used and correct it a dozen times.

Don't hardcode elements that may change. If you want to provide users with a list of attorneys in the office, don't hardcode this information into a multiple choice variable that will need to be modified as attorneys come and go. Instead, in HotDocs, use a "answer source" that can be later

modified without changing the template or associated component file. Such "designing for maintainability" is an important development practice.

Use repeating lists. Do not create variables like Incorporator1 and Incorporator2 where there could be an unlimited number of incorporators. Instead, use repeating lists—"repeats" in HotDocs—that allow the user to specify on the fly how many incorporators there are, and keep track of their respective attributes with one single underlying set of variables.

Use data validation tools. For example, if the address for service of process must be in the same state as the state of incorporation, test for this condition and do not accept data that violates this condition. Where exceptions are allowed, provide a user warning.

Name variables consistently. Develop a scheme for naming variables and stick to it. Don't use "Name of incorporator" in one template and "Incorporator name" in another template. Otherwise the user will be asked twice for the same information. Remember that HotDocs is case sensitive, so follow consistent capitalization conventions. And clean up after yourself: if you no longer need a variable, delete it so it that it does not confuse you or others later on. HotDocs provides a valuable Template Manager feature that can help you identify unused variables across an entire set of templates.

Think through your "component file" strategy. If you are using HotDocs and your initial system has more than one document, you will need to decide whether to use one or more component files. In HotDocs, each template has a component file with an associated list of variables, dialogs and related information. Documents can share a single component file and this often makes sense if they all flow from the same facts and decisions. For example, if you are producing a certificate of incorporation and a set of bylaws, and there are sufficient shared variables that will be organized into the same dialogs, you may want to point to a single component file. But if you want the same questions organized in different ways for each document, some extra steps are necessary with a shared component file, and if you need to have different developers working on templates in the set at the same time, a shared component file is counterindicated.

Design readable templates. Make your document templates as easy to read as possible. This will make it easier for developers and legal experts to review them. If your software lets you insert comments, indents, and spaces that won't show up in generated documents, use these features to clarify the template logic. HotDocs recently added the ability to add comments to templates.

Test frequently and backup often. Test and correct small pieces of your templates at a time in order to simplify debugging. And backup your files often to minimize loss of your work through some hardware or software mishap.

Minimize interruptions. Document automation work requires careful attention. Arrange to work when and where you won't frequently be interrupted by phone calls or impromptu visits.

Stage 8: Testing and Refining the System

Before you roll-out your system, test it thoroughly as a whole. Create many different answer files that test different permutations of data. For example, does the system work properly if there is only one incorporator? For "C" as well as "S" corporations? Ask others besides the template developer to test it as well. Someone not as close to the project may be able to spot weaknesses in usability that will not be obvious to the developer. Be a fly on the wall and watch people as they work with your templates, asking them to think aloud, but resisting the urge to intervene.

Stage 9: Writing Documentation

Documentation, both for the user and the developer, should not be viewed as an afterthought, but as an ongoing dimension from start to finish. Dispensing with careful documentation should only happen as an explicit economic decision. There are several varieties of user-oriented and developer-oriented documentation.

Make help available within the system. On-line, context sensitive help is often the most useful form of user documentation on both technical and substantive legal issues. Use the software's own features to build help screens or reference external help files.

Develop a user guide. A printed guide should provide guidance on when and when not to use the system. It should also have screen shots that highlight key decisions that users must make, along with explanatory material. Include a frequently asked questions section.

Maintain design documentation. If you agonize over a design choice, record your considerations while they are still fresh in your mind. This will spare you and others from having to resurface them —or worse, forgetting them—when a similar issue returns. We keep a record of major design choices in our "action plan." Sensible record keeping is called for, not compulsive standards writing.

Annotate your code. Templates and code that make sense to you now may become inscrutable later. Jot down some explanation while you have it in mind. In HotDocs, computations and scripts can be commented by inserting material after the word QUIT or double back-slashes. It is also possible to place comments in the templates themselves that will self-destruct when a document is assembled.

Print a list of variables and other system components. HotDocs Pro has the built-in capacity to print out a list of variables, dialogs and all associated scripts. This is a quick way to document the current state of your system during development and should be made a part of a technical documentation binder at the end of the project.

Stage 10: Rolling-out the System and Training Users

Start with a pilot system. Rolling-out a pilot or "beta" system to a select group of users makes a lot of sense. Solicit feedback and correct all discovered bugs—both technical and substantive—before rolling-out the system to the entire corporate department.

Provide training. Don't simply install the system on user desktops. You must provide training. Small hands-on training classes (no more than 6-8) work well. Provide one-on-one training to those that need it. And remember, training is an art. Engage a professional or at least someone with positive experience.

Establish answer file naming standards. Be sure to instruct users on how to name answer files, the files containing matter-specific data. The naming convention should make answer files easy for others to locate, retrieve, and modify. For example, in the incorporation system, you could use the name of the client/corporation followed by the name of the lead attorney, such as "Widgets Inc. – Smith."

Stage 11: Maintaining the System and Supporting Users

Few things will ruin a successful system more than failure to maintain it. Both the content and the technical aspects of the system require proactive maintenance. A single person should take primary responsibility in each area. If the corporation code changes, be sure to immediately make any relevant changes to the system. Users need to be able to rely on the substance behind the system. On the technical side, resolve problems as soon as they are reported. Nothing turns users off more than annoying bugs that are not quickly fixed. Finally, provide a help desk or other clear means of support. Users need to know who to contact with questions—both technical and substantive. Circulate news about updates and "tips and tricks" via an in-house newsletter or e-mail distribution list.

Document assembly systems are fragile things: without regular care and feeding they can easily spiral into disrepair and disuse.

Stage 12: Bringing out Version 2

Keeping it simple, in part, means leaving plenty of room for new features in Version 2. Survey users to find out what they like and do not like about the system. Keep a running list of features to add and improve. Don't let the system stagnate but again, be reasonable in what you promise and when you promise it for. After a reasonable period of stability (at least a few months), kick off a secondary project to issue a new release.

Getting there is half the fun

Think ahead. All things in moderation. To thine own self be true. Eat your Wheaties.™

It's easy to lay out platitudes about legal automation projects. But most real projects are messy affairs. Some flow smoothly to great results; some get there eventually after lots of detours and wrong turns; others stumble along and never seem to finish; yet others crash and burn. (The average failure rate for software projects across industries has been estimated as well over 50%.) Bad things happen to good people, and, contrariwise, sometimes folks achieve successes they don't quite deserve. But if you keep in mind the five key ingredients of a successful document assembly project—concept, people, tools, knowledge, and process—and keep alive an active spirit of design and planning, you will increase your odds of coming out ahead.

May you enjoy the ride as well as the destination.

==========

Marc Lauritsen (marc@capstonepractice.com, 978-456-3424), a Massachusetts lawyer and educator, is president of Capstone Practice Systems, a firm that helps professionals work more effectively through practical knowledge systems.

Alan Soudakoff (alan@capstonepractice.com, 781-925-1573) is vice president of Capstone Practice Systems. He is a member of the New York bar.

Sidebar 1: Viva la Difference

No two document assembly projects are the same, even within the same organization. Here are some of the ways projects differ. Where does yours fit?

Users	Who are the intended users of the application? Lawyers, paralegals, secretaries, students, clients? Are they experts in the area in question, or novices? Are they a few or many? Do they work in proximity, or are they spread among floors, offices, or cities?
Documents	What documents is the system designed to produce? Short and simple, long and complex, or somewhere in between? Are they typically first-draft-final, or do they require lots of post-assembly editing? Are official, graphical forms involved? Are the documents typically produced individually, or in related sets? Can they be neatly handled with fill-in-the-blanks variables and alternative/additional passages, or do they involve lots of material that doesn't lend itself to straightforward rules?
Scope	What range of transactions is the system intended to support? How deep do you intend to go in modeling the variations from transaction to transaction? Is the system designed only to produce first drafts, or guide users through several stages of revision and negotiation? Should it offer project management and decision support features?
Purposes	What are the driving goals of the application? To speed up processing? Improve quality or consistency? Achieve greater capacity? Allow work to be delegated to more efficient staffing levels? Assist in training?
Novelty	Is this the organization's first effort of this kind, or one among several? Are the team members experienced in this kind of thing? There are vast differences between doing a first project of this kind and later ones. They involve different states of organizational and personal readiness.
Staffing	Is the project being done entirely with in-house personnel, by an outside consultant, or some blend? Is it conceived as a project by lawyers in a practice group, aided by others, or as an initiative of the IT department, aided by lawyers?

Sidebar 2: Illustrative time commitments for an 8-week project

Roles and Stages								
	Defining & Designing (Stages 1 and 2)	Planning (Stage 3)	Settling Document Models (Stage 4)	Building the Technical Environment (Stage 6)	Programming and Testing the System (Stages 7 and 8)	Writing Documentation (Stage 9)	Training and Roll-out (Stage 10)	Maintaining and Supporting (Stage 11)
Legal expert(s)	M	M	H	-	L	M	L	L
Project manager	H	H	M	M	L	M	H	M
Template developer	M	M	L	L	H	M	L	M
"End users"	M	M	L	-	L	-	H	L
IT professional(s)	M	L	L	H	L	L	H	L
Trainer	-	-	-	-	-	L	H	L
Time frame	Week 1-2	Week 3	Weeks 1-3	Week 4-5	Weeks 4-6	Weeks 3-7	Week 8	Ongoing thereafter

Code: **H**igh - One day or more per week; **M**edium - a few hours a week; **L**ow - a few hours a month.
Note: The roles listed here are sometimes shared and often overlap, as do the stages of the project. For the purposes of this chart, it is assumed that Stage 5, Selecting the Software, has been done in advance. And our Stage 12, Bringing out Version 2, of course only really begins once the initial project is over.