

TEMPLATE DESIGN TRADEOFFS ENCOUNTERED IN CREATING AN AUTOMATED COMMERCIAL MORTGAGE LOAN DOCUMENT ASSEMBLY SYSTEM

By: Gary Whittington
Assistant General Counsel
AEGON USA Realty Advisors, Inc.
4333 Edgewood Road, N.E.
Cedar Rapids, Iowa 52499-5223
319.398.8633
gwhittin@Aegonusa.com

Commercial mortgage loan document assembly systems promise to reduce significantly the time spent adapting basic commercial lending forms to facts and circumstances presented by a specific transaction. Technophobes take comfort - no commercial loan document automation system will produce perfect documents. At best, they will produce very good working drafts quickly.

Whether a PPD (normally an abbreviation for "prospective prom date", but in this case, "prospective project designer") knows it or not, he or she will be making early decisions important to the project's success. How smart should the templates be? Is the system's purpose primarily document assembly or knowledge management? Who should be on the template design team? If the team members are lawyers and IT professionals, will risk aversion reach levels toxic to project completion? How much consulting time should be used? How much testing will be performed before the system is placed in service? How close to perfect should the output be? Will modifications to templates be made continuously, or will periodic revisions be published following peer review and testing? How quickly will the project need to be completed in order to achieve functionality before obsolescence makes new revisions necessary?

There are no right answers. Experience teaches that tradeoffs will be faced as the project unfolds: the PPD will need to make intuitive judgments. I hope that this brief discussion will give a novice template developer an awareness of some of the tradeoffs that he or she will face, so that these decisions he or she makes can be considered in advance and made as wisely as possible.

I. KNOWLEDGE MANAGEMENT VS. DOCUMENT ASSEMBLY

The knowledge gathered and keyed into a computer can form the basis of a relational database for future use, or can be keyed in once for a single document assembly and

stored in static answer files. A database is obviously superior, if the information has future utility for knowledge management, document assembly, or reporting purposes. The infrastructure, however can be costly. What internal programming resources are available and at what expense? How will data integrity be achieved and maintained? This decision will hinge on institutional or client willingness to dedicate resources and time, and on the future value of the information.

II. FULL DOCUMENT AUTOMATION VERSUS GRAPHIC FORMS

Full document automation creates ordinary documents that can be manipulated and edited using the underlying word-processing software. Since this is the first form of document automation that appeared on the scene, it is easy to overlook the fact that for a new project creator today, the threshold question is whether to use a text form or a graphic form with text boxes. A graphic form is analogous to a printed form, but with defined text boxes completed by the user using the full capability of the automation software. Thus, within the text boxes, document automation is developed using rules and variables. Outside them, the document is static and no revision is possible.

Deciding whether to use graphic form automation depends on the same factors that would influence the decision whether to use a printed form or a word-processed document. An obvious advantage is the complete reliability of the output within the graphic area. A reviewer of a document who is familiar with the underlying form can focus entirely on the portion of the document within the text windows.

Although the author confesses to inexperience in the use of graphic forms, the relative difficulty of creating and editing them is readily apparent. Extremely detail-oriented manipulation of scanned forms is necessary in order to replicate them in electronic form, the starting point for automation within text areas. However, on a cheerful note, no reasonable human being could expect a lawyer to perform the task of perfecting the graphic portion of the form.

The complete standardization of selected portions of the document, coupled with the availability of automation in the document portions, may be attractive when a user group operates outside organizational controls, or in market niches where standardization is highly valued, such as conduit lending.

It may be hard to distinguish graphic clauses from those that should be made dynamic in text boxes. Poor decisions may lead to conflict as borrowers insist on revisions to graphic sections. If an institution can

tolerate, as a stylistic matter, riders that renounce the contents of the form ("Notwithstanding anything to the contrary in the foregoing . . ."), this problem can be solved. However, if much of this takes place, one may legitimately question the overall success of the project in terms of document coherence and quality.

III. DYNAMISM VERSUS CLAUSE MANAGEMENT

Another major decision, which will affect text automation in either of the full automation or the graphic environments, turns on the tension between dynamism and clause management. It is simple enough to cause specific facts to be imported into a template. The more daunting challenge is to decide how underlying text will be modified based on rules. This is the seductive lure of artificial intelligence (or artificial negligence). How smart do the templates need to be to achieve the most efficiency at the lowest cost? Remember the 80-20 rule. Your client will when he or she gets the bill.

The flow of automation in a dynamic system emulates the thought process of a draftsman who is writing a document based on a form, making decisions on particular modifications based on rules. The flow of automation in a clause management system is "chunkier." Rather than attempt to capture the logic underlying each drafting decision in clauses that would require complex or extensive automation, the project creator opts to design a base template into which clauses including such variations will be inserted. The project creator may implement this feature either through relatively simple rules incorporated in the master template (usually as subtemplates) or through requiring work from a human draftsman following assembly of a base destination document. In the latter case, the automation software can be used to facilitate clause insertion.

What are the costs and benefits of each of these approaches?

The dynamic approach creates a template that can quickly grow opaque. Rules proliferate quickly and numerous interrelated rules create complex logic. Complex logic, while intriguing, can lead the project creator into drafting tasks which may, in hindsight, prove not to have been cost effective. However, such a system disciplines the draftsman to know exactly what he or she is doing, and to push the capability of automation to its practical limits. It may be more expensive on the front end, but if well done may reduce assembly time or revision time on the back end. It will be more prone to mistakes arising from faulty programming logic (whether because initial drafting assumptions are later disproved, or because complexity

begets error). A dynamic system is more appropriate to an organization with a single "czar," since a higher level of expertise in the use of automation software will be required.

A clause management system is easier to understand and is less highly automated. For example, such a system might import into the base document, under a relatively simple set of conditions, a clause that is intended only to be the most desirable starting point for a manual markup. Alternatively, numerous clauses may be made available to a human draftsman, requiring a manual choice. This choice may be forced during assembly of an initial template, or may be handled through the preparation of a library of clauses available during document revision. Such an approach may result in a need for more frequent conforming changes to parallel clauses in order to implement general changes in policy or practice. Even so, the project creator can achieve significant automation through internal automation of clauses. Another advantage may be the decentralization of maintenance responsibilities. Automation of clauses to incorporate a few text or multiple choice variables is a simple task. Under this model, more draftsmen can be drafted (read, "suckered") into the task of maintaining the template set.

In general, the clause management approach is only necessary in cases where a proliferation of rules renders the automation logic unduly cumbersome given organizational conditions, or where the project creator believes that the need for a judgment call by a human draftsman is inescapable (because the best underlying rule is still an oversimplification).

At particular points in the automation process, the project creator should understand which of these approaches he or she should favor, based on the nature of the client and the composition of the development team. If the project creator decides to use the software to design a clause management system, some of the drafting power of the software as a vehicle to enforce uniform application of rules will necessarily be lost. The decision may still be rational, however, if human or financial resources for development of a more sophisticated system are scarce. As a practical matter, most project creators will find that they have designed, in the end, a hybrid system incorporating each of these approaches under appropriate circumstances.

IV. PERFECT AUTOMATION VERSUS EVOLVING FIRST DRAFT

Good lawyers are drilled to become perfectionists from the moment they emerge from the proverbial egg. When undertaking a document automation project, however, it is

necessary to make a strategic decision concerning the degree of perfection that makes economic sense under the circumstances. Perhaps the most important circumstance is the number of documents to be created from the templates. When a document will be created only a few dozen times in a year, it should probably be less highly automated than documents that will be assembled by the hundreds.

The relationship between the work done in template automation and the degree of perfection achieved is like a curve that approaches but never reaches the axis. This observation may seem obvious. But the project creator should always be alert to the possibility that excitement over his or her developing craft may lead to overly fussy attention to the automation of details which might have been more easily addressed through conventional boilerplate drafting techniques.

There will always be anomalous transactions for which extensive custom drafting is required. Perhaps these will arise because of the negotiating leverage of borrowers or because of peculiar characteristics of a particular real property, but arise they will. Therefore, at some point, the template developer will have to draw the line and declare that a template or system is good enough.

This implies that human beings will need to be responsible for reviewing and modifying the first drafts spun off by the templates. But such persons, who may not be entirely familiar with the logic underlying the project creator's rule decisions, may become intimidated. This may lead to a view that the output of templates is sacrosanct. In such an environment, the project creator may lose an important opportunity to build a feedback loop for template correction and evolution.

Decisions on where to draw the line and start using the templates should take into account the volume of a template's production, the expense of further automation, and the sophistication of the users in making revisions.

V. FIAT VERSUS CONSENSUS

Unless the design project is relatively simple, the project creator will need to make numerous decisions during the design process. These decisions will directly affect the functionality of the templates and the quality of their output. Designing the templates in solitude or as part of a small team is efficient and imposing them on users by fiat is extremely efficient. But if no one wants to use the templates when they are finished, what good are they? Consensus and "buy-in" are optimal, but building them takes time and costs money. Those with the most valuable input

may be the busiest and the hardest to attract to frequent meetings.

The most rational decision in this area will necessarily depend on particular institutional conditions. If there is a broad consensus on standard institutional practices, the task will be easier. However, it may be that the template logic itself will become an expression of standard policy, or that the channeling of the institution's attention to those decisions may raise institutional consciousness concerning inconsistent lending practices. Unless practices are uniform among users, the project creator may need to create additional user options, in order to prevent the project from stalling and achieve "buy-in."

One model that may reduce the consensus requirement is to make the use of templates optional, and to make the project creator responsible for attracting users through the quality of the templates. This can be a good tradeoff for the project creator, if he or she has a good enough feel for the document set to get close to the mark alone.

VI. PARAGON VERSUS TEAM

The ideal document automator would be a perfectly expert mortgage loan attorney who is also perfectly proficient in the use of document automation, but whose time is not very valuable. Obviously, we are in quest of a mythical beast, rarer even than the client who would pay a law firm by the hour to complete a full blown mortgage loan document automation project. While awaiting the appearance of the perfect automator, the project creator may need to stitch together an imitation from available components. These will include ordinary lawyers and IT professionals, who will need to work together closely and cooperatively.

The lawyer on the team will have to open his or her mind and accept certain programming disciplines that may seem tedious, but which an IT professional will understand as essential in the long term. For example, variable-naming conventions may seem initially unnecessary and result in some cumbersome names, but bear fruit in overall transparency as the variable set expands. The IT professional will need to accept that excessive documentation and testing in templates may unduly delay or kill the project. The risk that a template fails to behave perfectly, after all, is merely the risk that we have to prepare the document as we have for our whole professional lives.

CONCLUSION

The project creator should be visionary, keeping fresh in his or her imagination the prospect of pushing a few buttons and having a full set of perfect mortgage loan documents appear, tailored to the transaction and formally perfect (even though this will probably never happen). If the template project is ambitious, even reaching the point where the templates work reasonably well is not too easy and will take some time. The novice project creator may not realize that he or she will be drawn into issues of system design and computer programming. The genius of the most widely accepted document automation software is that it may entice a lawyer into this unfamiliar realm with its seemingly simple user interface.

Remove all sharp objects from your office. Keep your sense of humor, and be warned that the final prize may at times seem as distant. Hopefully, these descriptions of a few likely crossroads at least help the PPD stay pointed in the right direction for his or her particular project.

Z:\DMiddaug\DMiddaugh\Docs\Gary\Miscellaneous\Template Design - ABA.doc