Current Frontiers in Legal Drafting Systems

Marc Lauritsen
Capstone Practice Systems
193 Bolton Road, Harvard, Massachusetts 01451
marc@capstonepractice.com

[A Working Paper for the 11th International Conference on AI and Law, Stanford University, June 2007]

ABSTRACT

Most legal tasks involve document preparation. Drafting effective texts is central to lawyering, judging, legislating, and regulating. How best to support that work with intelligent tools is an ancient topic in AI-and-Law circles. This article surveys the history and current state of document drafting software and associated theory. Present frontiers in both of those fields are identified, and a preliminary sketch is made of a 'grand unified theory' of legal drafting systems.

1. INTRODUCTION

1.1 A note on vocabulary

Readers will note varying terminologies in what follows.

"Document assembly" has most commonly been used to refer to the technologies discussed here. I have always found that phrase regrettable, since it really only describes one particular *function* that drafting systems can perform. Such systems can and should provide knowledge-based support for organizing facts, research, analysis, and other processes as well as specific documents. Also, it implies a *mode* of use in which textual components are combined automatically based on user responses to extradocumentary queries, whereas other more directly interactive modes are increasingly supported by the technologies in question.

"Document modeling" fails to capture the *use* of models.

"Document automation," on the other hand, suffers from overbreadth. It sometimes refers to complementary technologies such as document management, comparison, and analysis tools, and word processing features like automatic numbering and cross-references. And it again implies a degree of automaticity that doesn't comport with the mixed initiative nature of present day and emerging drafting technologies.

Computer-aided, advanced, or intelligent "drafting" systems (in the sense of composing texts) probably come closest, encompassing preparation of legislation as well as legal instruments.

1.2 Basic concepts[1]

Computer-aided drafting software is reasonably common in the contemporary legal world. A lawyer, paralegal, secretary, or do-it-yourselfer works through a series of question/answer dialogs, perhaps laced with reference material, and the system assembles a draft document. Or the user picks forms, clauses, and other components as needed from libraries of options and alternatives.

[1] Parts of this and the next section are distillations of [17].

Sometimes such an application is obtained from a legal publisher or software vendor. (TurboTax from Intuit is a familiar consumer example in the United States.) That brings the benefits of automation with little effort and expense. Sometimes an organization develops a custom system with a document assembly "engine," using its own forms and experience. That can require a fair amount of up-front time (thinking through and handling many possible scenarios), but can result in excellent leverage of practical legal knowledge.

Such systems capture regularities underlying the documents – what sections, paragraphs, sentences, and words go where under what circumstances. The software prompts you to make choices and specify details like names, numbers, dates, and phrases. Instead of cutting and pasting, you pick desired options or alternatives from lists. Instead of searching for phrases like "Lender's name" and replacing them with your client's name, you respond to questions and let the computer do the needed work. Instead of keyboarding lots of text and fussing with formats, you let the application handle all the predictable variations, boilerplate, and layout.

Terminology varies among programs. There is usually a "template" that models a particular kind of document, with variables and instructions placed at locations that need to change from case to case. You answer questions in a series of interview-style dialogs, the responses are stored in an "answer file," and the desired document is generated in a common format like Word, WordPerfect, RTF, or PDF. [Most programs do not yet directly support Open Document Format (ODF).]

Typically each answer file stores all the data relevant to a single client or matter, and thus can be used to generate more than one document or form (e.g., a complaint for divorce, a financial statement, and various motions in a family law system). When answers are changed, the documents can be instantly re-generated.

In addition to basic point-and-shoot clause selection and fill-in-the-blanks variable replacement, these systems can store drafting rules and practitioner know-how that guide the hand of novices and experts alike. For example, a divorce system may ask about the client's state of residence, financial situation, and number of children. Then, based on answers to those and follow-up questions, it will insert appropriate material in the complaint and associated motions.

2. APPLICATION SOFTWARE

2.1 Word processing beginnings

Document assembly systems had early echoes in the search-and-replace, macro, merge, and related features of word processing programs.

Search-and-replace functions allow users to locate all instances of a given word, phrase, or string of characters and replace them with another word, phrase, or string. A boilerplate document with placeholders like [plaintiff], [defendant], [court], and [attorney for plaintiff] can thus be tailored for a given case by replacing those phrases with specific information. These replacements are ordinarily done one at a time, although they can also be under the control of a macro.

A *macro* is a series of recorded commands that can be played back when desired. Macros can retrieve documents, pause for user input, call other macros, and do anything a computer user can do from the keyboard.

A *merge* involves the combination of text from multiple files. An elementary use (the so-called 'mail merge') is creating a series of customized letters by inserting addresses, salutations, and other information about different people (contained in a 'secondary merge file') into a boilerplate form (a 'primary merge file'). Merge routines in modern word processors can prompt the user for data and launch macros.

Most word processing programs have long supported explicit programming with conditional branching – i.e., the inclusion of decision points at which alternative procedures can be followed depending upon information up to that point, using IF, THEN, ELSE commands and logical operators like AND, OR, and NOT.

By chaining together macros and merges, and taking advantage of common word processing features like automatic paragraph numbering, it is possible to construct quite satisfactory systems for automatic assembly of moderately complex documents. But hard-to-maintain "spaghetti code" often results.

2.2 Specialized programs

Specialized programs for legal document assembly emerged in the late 1970s and early 1980s. The ABF Processor (developed by James Sprowl at the American Bar Foundation [23]) and CAPS (developed by Stan Neeleman, Larry Farmer, and Marshall Morrise at Brigham Young University) were two of the earliest research efforts. Commercial products like Document Modeler, WorkForm, Work Engine, DocuMentor, FlexPractice, ExperText, and Scrivener soon followed.

These applications offered advantages over word processors, such as:

- far greater ease of authoring, maintenance, and distribution of models
- nicer interface for data entry and user guidance
- support for graphical forms
- easy re-use of data across sessions and templates

The same automation of routine text editing processes that made word processing so pervasive thus became attainable on the conceptual level of document assembly. If word processors can be thought of as helping you edit text with power steering, using advanced drafting tools is like being driven in a (robotically) chauffeured limousine.

Current document assembly programs use a wide spectrum of approaches and boast an impressive array of innovative features. They excel in the richness of their user interfaces and the sophistication of documentary output. See Section 2.7 (*Commercial Engines*) for some of the choices and considerations.

2.3 Important differences

Word processing documents vs. graphical forms. Document assembly generally encompasses both freely editable word processing documents and fixed-format, "graphical" forms, where the background is static and information can only be placed in pre-designated fields.

Questioning and guidance vs. document generation. In most document assembly applications users provide information and make drafting decisions through questionnaire-like screen dialogs that are outside of a target document. There is a discrete interface through which questions can be asked and guidance can be given. Many document assembly tools can in fact be used to produce information gathering modules, advisory systems, and intelligent checklists that needn't result in any document at all.

Professionals vs. self-help users. Document assembly technology can be and is being used both by professionals serving clients and by individuals doing work for themselves. There are also hybrid scenarios in which the client completes a computer-based questionnaire on his or her own, and the answer file goes to the legal professional for further review, revisions, and document drafting.

Users vs. developers. Document assembly software typically involves distinct tools and interfaces for end users and developers. Many features and issues that are critical for people charged with developing applications are irrelevant to the ultimate users, and vice versa. Some software choices offer excellent end user interfaces but clumsy development tools, and vice versa.

2.4 Online document assembly

The World Wide Web opened up new opportunities for organizing and delivering document assembly applications. Any or all of the major components – the engine, templates, answers, documents, help material – can be served from a Web server, providing location independence, multi-user access, *client* access, ease of use, and other benefits. For law office staff, a big advantage of Web-based implementations is the centralization and instant updating of template collections. Access to robust document automation can be delivered without special purpose local software having to be purchased, installed, configured, and maintained. Often just a browser is required, together with an Internet connection and a printer. For IT professionals (and budget conscious managers) a single centralized server and staff can economically provide document assembly capabilities to hundreds or thousands of users.

Nonetheless, there are still advantages to desktop (full local processing) modes. For example, built-in library interfaces for choosing templates, in-context data entry and revision for graphical forms, the ability to assemble ad hoc combinations of documents, out-of-the-box database connectivity, clause libraries, easy local customizations of templates, and the ability to function while disconnected from the Internet.

2.5 Varieties and venues

Document assembly technology has been applied to everything from simple thank-you letters to elaborate expert systems that

advise on the laws of many jurisdictions and generate document sets reaching into hundreds of pages. There is a vast range of application types. The main polarities include: off-the-shelf vs. custom-built; in-house vs. client-facing; textual vs. graphical; question-driven vs. clause-selection; desktop vs. online; document-oriented vs. interview-focused. The combinatorial possibilities are staggering.

There's likewise a great variety of *contexts* in which document assembly is used.

Small law firms and legal departments most commonly use document assembly for routine or high-volume paperwork, purchasing pre-written template sets when possible. **Larger firms and departments** are more likely to develop custom in-house applications, drawing upon their own precedents and integrating with knowledge management efforts. But some practitioners in both settings are increasingly interested in sophisticated, high-end drafting applications that combine advanced models of complex documents with rich layers of annotational guidance. And firms of all sizes are experimenting with outward-facing applications aimed at clients and non-client customers.

Corporate law departments are starting to show great interest in document automation for client self-service. Cisco and Microsoft, for instance, now provide do-it-yourself sales contracts, non-disclosure agreements, and software licenses to their business users. Law departments can off-load routine work and delight their clients with rapid turnaround, while retaining control of transactions that vary from pre-approved, "safe" terms.

In the **nonprofit legal services** world there have long been initiatives, on- and off-line. Some legal aid organizations have developed their own systems and made them available to fellow programs. An example is Greater Boston Legal Services, whose family law and eviction defense systems are starting to be used widely by advocates in Massachusetts. The California-based I-CAN!™ (www.icandocs.org) project has served interactive forms to thousands of lay users over the past several years. "National Public ADO" (Automated Documents Online – www.npado.org) is a related effort, funded by grants from the federal Legal Services Corporation and software donations from LexisNexis.

Courts have taken a strong interest in automated forms as a response to the deluge of self-represented litigants. Many state court systems have made their standard forms available as fillable PDFs, and several (including California, Idaho, and Utah) have mounted much more sophisticated, interactive applications. The integration of document automation and e-filing raises especially interesting possibilities. (See Lauritsen & Janis 2004 [12].)

Then there are the **commercial** players that target the consumer marketplace. There are now many fee-for-service providers of pre-fabricated forms aimed at self-helpers. See for instance www.uslegalforms.com, www.smartlegalforms.com, and www.legalzoom.com. Plus there is a growing universe of non-lawyer "legal document preparers." (See, for example, www.wethepeopleusa.com and www.naldp.org.) These private sector developments are helping to expand consumer choice – and shake up a complacent legal profession – but may pose questions of second-class quality, especially for disadvantaged citizens. They also do not provide the insurance function of a lawyer (someone to sue if things go wrong.)

We're beyond these applications simply operating as power tools in the hands of skilled professionals. Increasingly they are being used directly by consumers. People have long done their own wills and taxes with off-the-shelf packages. Now large international law firms sell subscriptions to online expert systems that deliver sophisticated legal analysis without direct human involvement. Corporate law departments equip field personnel with do-it-yourself contract assemblers. Courts and legal aid programs provide intelligent forms for unrepresented litigants. And lawyer-less entities vend interactive documents and automated legal assistance over the Web.

Document automation has steadily gained traction in law. We may be entering a period of faster growth. The signs include vigorous competition among strong vendors of excellent products, a large community of qualified consultants, rising expectations from clients, aggressive new competitors giving consumers alternatives to the profession, and huge latent opportunities for process improvement in legal work.

2.6 Professional reception

The craft of lawyerly drafting does not immediately resonate with advanced technology. Drafters today have come to appreciate the power-steering aspects of modern word processors, with cut-and-paste, spell checking, search-and-replace, auto-numbering, and similar features. Being able to revise drafts on the fly, insert extended passages with a couple keystrokes, and quickly manipulate their format and structure are welcome powers that are now largely taken for granted. Word processing for the most part is perceived as a power tool in the hands of a craftsperson.

But technologies that involve more autonomous software behavior are unsettling to many. Documents that automatically fill themselves out, or populate themselves with relevant provisions as users interact with them, can be downright scary. Lawyers may not be accustomed to working at such speeds. And the whole concept of interacting with a document under draft via an extrinsic "interview" can seem an unwelcome distancing of the drafter from her work.

In part, this is a matter of getting used to a new method of wordsmithing. Once understood, intelligent document assembly can be every bit as power-tool-like as word processing. The drafter makes decisions and specifies information in an interface that lives above and apart from the words themselves, with confidence that the right stuff is going in the right place, and great freedom to recast large segments of the draft by simply toggling switches. Lawyers who understand the general operations of a document assembly program, and the specific logic embedded in a particular model, can achieve both high efficiency and good professional satisfaction.

New pressures for document quality and auditability arising from regulatory compliance and ethical concerns will drive legal and business professionals from today's artisanal approaches to drafting toward systems-engineering ones. The business process change management challenges involved will be daunting for many organizations.

2.7 Commercial engines

There has been a dizzying variety of technologies, vendors, and approaches in the legal document assembly universe since the late 1970s. In a recent exercise I was able to list sixty-five discrete engines aimed at lawyers that have been commercially available at some point. (Most are long gone.)

Alan Soudakoff and I did the last comprehensive public roundup in *Law Office Computing* ("Shopper's Guide to Legal Document Assembly," [19]). This *Consumer-Reports*-style analysis of ten leading products covered several dozen of the most important comparative features. In more recent private analyses we've identified hundreds of differentiating characteristics.

Here is a very brief and subjective sketch of today's leading candidates.

HotDocs from LexisNexis (www.hotdocs.com) has the biggest market presence and most developed ecosystem. It has an excellent online knowledgebase, email discussion list, and consultant community. HotDocs offers the best tool for automating graphical forms, and has a full-featured Web implementation. The company continues to release significant new versions each year.

DealBuilder from Business Integrity (www.business-integrity.com) is purely Web-based on the user end and offers an AI-based authoring environment that reduces the need for traditional template programming. Precedents that are marked up in ways intelligible to substantive experts can often be converted automatically into interactive "masters." Business Integrity established a beach head in the London Magic Circle firms, and has made major inroads into top law departments there and in the US, building on the self-service themes mentioned above.

GhostFill (www.ghostfill.com) is a vigorous player from Korbitec in South Africa. It was integrated into the Amicus Attorney case management software, branded as Amicus Assembly. It also underlies the new and improved construction contract software from the American Institute of Architects and Drafting Wills and Trusts from West Publishing. GhostFill has a programmer-friendly object-oriented and open architecture, making it very easy to add functionality. It offers great flexibility for custom integration, and can be easily hooked up to databases out of the box.

Rapidocs (www.rapidocs.com) also originated in the United Kingdom. It includes innovative features that optimize it for ecommerce applications. It's also active in the non-lawyer space – see www.directlaw.com.

Exari (formerly SpeedPrecedent), from Exari (www.exari.com), is a web-based solution based in Australia with a strong commitment to open systems and standards, especially XML.

QShift from Ixio Corporation (www.ixio.com) is an Internet subscription-based application with the slogan "smart document drafting on demand." I think of it as a clause manager on steroids. It has powerful underlying technologies that can take it in many different directions.

D3 (Dynamic Document Drafting) from Microsystems (www.microsystems.com/d3) is one of the latest entrants. It has broken new ground in terms of tight integration with Microsoft Word (2003 or better). While possibly weak on some of the more advanced aspects of high-end document automation such as multi-level repeats, D3 includes styles management, group security, and collaborative authoring features that aren't seen in most other products.

Other active players include **Perfectus** (www.perfectus.net), **ActiveDocs** (www.keylogix.com), and **Pathagoras** (www.pathagoras.com). And even though they are no longer marketed or supported, quite a few offices still use old stars like **CAPS** and **PowerTXT**, or have recently converted from them to contemporary platforms.

There are of course many relevant software alternatives *outside* the specifically legal context, such as **Schema** (http://www.schema.de/eds/en/), that legal drafting technologists would do well to study.

3. APPLICATION FRONTIERS

Almost every imaginable document assembly feature has been engineered by someone, and the leading products easily cover all the essentials. The technology is way ahead of what most users are doing with it. Yet major dimensions of evolution remain (some previously seen in software species now extinct.)

3.1 Longitudinality

Lawyers have long requested support for automated documents that can re-assemble themselves (e.g., based on changed deal characteristics) after a user has edited them (e.g., to reflect negotiated language adjustments). Right now, most engines are optimized to produce first drafts. The output document retains none of the template's intelligence. The wish is for that intelligence to survive and be invocable *throughout the life of a transaction*. I've started to refer to this as "longitudinality." Others call it "round tripping," referring to models that somehow survive the transit from one software environment to another and back, or one user to another user and back.

Several past engines, notably thinkDocs and SmartWords, came close to this. D3 from Microsystems is one of the first new products to tackle them. I'm aware of comparable plans by other vendors.

A related notion involves support for "contract lifecycle management." Facts entered and decisions made in the drafting stage can and should have significance for the later disposition and management of a document, such as who gets to do what with them and which of the obligations and prohibitions they embody need to be tracked.

New drafting environments will blur the distinctions between document *generation* and document *management*. Contracts will increasingly be manufactured for manageability. Decisions and data used in the drafting process will remain associated with the document for purposes of its downstream management. And considerations of manageability will be surfaced as part of the very drafting process.

3.2 Document as interface

Especially if people are able to continue to invoke the logical processing power of a template even after they have hand edited a draft assembled from it, they should be able to do so in a richly dynamic, intelligent, fully editable document right within a state of the art word processing system.

With seamless integration of assembly engines and word processors you can work *in* the latter as you make choices and enter information, and move easily between hand editing and auto-processing. The document itself communicates the variations it is designed to accommodate.

Our tools should accommodate both those who are comfortable working in an interactive environment and those who would rather quickly get to the basic word processor or paper (including those who prefer to mark up by hand).

Until recently, most advanced document assembly tools provided little interactivity between the question-and-answer activity and the assembled document. You did them in sequence, returning to the questions if the document didn't turn out as desired. People should be able to link between questions and the document contexts they impact. Answers should be revisable from the location of their impacts in the document.

Real time previewability of documents under assembly was present in some of the early document assembly engines, but only in the last couple years has it reappeared in a mainstream product. HotDocs for instance introduced the preview pane in its version 6.0, and has regularly extended its power. Now it's not only possible to see your document as it would be assembled given any configuration of answers, but (within some limits) you can also jump from a question to the locations in the document that it affects, and conversely, jump from an answer in the preview to the question that gathered it. Exari has similar functions.

Tools like Microsystems D3 take this one step further. There, drafters 'live' right within Word, with the questions presented in its task pane. You can seamlessly switch between hand editing the text and answering questions that trigger automatic changes. Several of the other vendors have similar mechanics in the works. Ideally the task pane will dynamically update itself based on your insertion point, showing the available parameters and other metadata relating to the passage(s) it is within.

To link questions with their documentary implications – and correlatively link document passages with their variable sources, if any – requires application-level knowledge of such connections and appropriate mechanics in the user interface. Such connections are relatively easy for merged variables, but more challenging for (1) conditional passages, including repeats and nested forms of both, (2) variables that have multiple impacts in a document or play roles in compound computations.

More ambitiously, the document editing environment would enforce models. Objects should not be allowed to be moved in ways that violate any structural/syntactic (occurrence, combinatorial, sequential, or subordination) constraints, unless the user explicitly overrides and creates an exception condition. A spreading flag could indicate which branches are logically resolved and which are not. Exceptions should also propagate up the hierarchy.

3.3 Flexible renderings

Word processors with graphical user interfaces long ago achieved WYSIWYG (What You See Is What You Get.) Unfortunately, What You See Is Not All There Is. Especially in intelligent drafting contexts.

Rendering can be understood as a projection of a higher dimensional structure onto a 2D surface. Beneath the 'surface' words there can be a lot of underlying substructure. Even the surface in today's word processors is more than just text. It includes all visible elements of a document (and some of its interactive functionality)

Users should be able to generate drafts both with and without commentary or other annotations. Document previews should render well even if no questions are answered. Color should be used effectively in both documents and interface.

Users should be able to toggle between a page-laid-out document and a view (full text or collapsible outline) that shows possible as well as actual instances of defined document objects. For instance, optional objects (and potential iterations) could be grayed out unless used, with identification of the condition(s) involved, and whether or not they are presently satisfied.

There should be some visual distinction between "not yet decided whether to use" and "decided not to use." A view mode should be available that suppresses non-used objects, with a distinction between choices that *were* made and choices that could have been made (and still can be).

Developers especially will find it useful to be able to modularize and de-modularize on the fly. In other words, flatten out a hierarchy of inserted subtemplates and computations into a single layer, effectively expanding/collapsing across such a hierarchy.

Markup in *assembled* documents is useful for much more than re-assemble-ability. Adding or leaving markup in assemblies supports uses outside the immediate drafting context such as workflow, document management, and data mining.

3.4 Role versatility

Another frontier involves enabling users to instantly switch between instance editing and model editing modes (or do both at once). Scrivener and Visual Workform were among the few historical products to dabble in that functionality.

Drafters ought to be able to model as they go, expressing such points as "this document needs to include section X" or "in the event this transaction ends up having this feature, be sure to include Exhibit Y." People should be able to articulate rules, in addition to just entering texts or adding commentaries.

One arena where role switching is handy is drafting work for which there is no pre-existing template. A draft might start out model-less, but gradually include modeling markup, and possibly give birth to models useful for later purposes.

3.5 Integration of commentary

Flexible management of commentaries and annotations, both during and after the assembly process, remains elusive.

Drafting systems should be able to deliver the whole spectrum of know-how expressed in pre-automation forms and commentaries. They should serve as intelligent gateways to model documents (forms) and associated commentaries, with as much interactivity and dynamism as people care to invoke.

You should be able to choose just to see the *commentary*, but also enter narrowing specifications, based on deal characteristics or subjects of interest, so as to create a more manageable, useful customization or subset of the commentary.

You should be able just to see the *form*, but also enter narrowing specifications, so as to create a more matter-specific starting draft.

If you choose *both*, you should be able to enter narrowing answers for both, and also choose how you want the two to be combined (separate assemblies, back to back in one physical document, commentaries in footnotes or endnotes, in-line commentaries in different color or font, or possibly even some table-ized side-by-side format).

The goal is to provide maximal flexibility to users as to how long to remain in the template and which of its features to use. Some people just don't like the idea of document assembly and may push to their word processor and/or paper early in the process. Others may choose to exercise a template's features in combination with an answer-set over the life of a transaction. People should be able straightforwardly to get artifacts closely resembling current forms and commentaries if they prefer not to work within interactive templates.

3.6 Collaboration

3.6.1 Collaborating with clients

Recent years have seen modest but steady growth in innovative legal service delivery methods based on distributed document automation systems. Some law firms have commissioned systems that package their expertise for use by personnel at client sites. Often these produce routine documents without further law firm involvement.

Most contemporary document assembly applications – online and off – still involve solitary users. One person at a time interacts with the application to enter information or generate documents. Sometimes people take turns working on the same matter or transaction: e.g., a lawyer has her secretary input basic data. Or a law firm lets its clients interact with intelligent questionnaires via an extranet as a way to reduce the cost of data gathering and to dispense background advice.

Even though there's little evidence yet of these technologies being put to use in modes where several people work *on the same task at the same time* – perhaps the clearest form of "collaboration" – the sequential, asynchronous lawyer/client example just given above is encouraging. Some theorists describe it as the "co-production" of legal work. It's straightforwardly done with several of the current Web-enabled document assembly platforms, including HotDocs, DealBuilder, and Perfectus.

In addition to private law firm extranets, co-production is happening now in corporate law departments that provide do-it-yourself contract assemblers for field personnel. And in non-profit, pro bono, and "low bono" contexts where people unable to afford commercial rate lawyers take advantage of "unbundled" legal services like ghostwriting.

3.6.2 Lawyer-lawyer collaboration

Document automation systems can also be engineered to facilitate cooperative work among lawyers, both within and across offices. For example, they can be modularized to allow specialists to focus on specific aspects of a large transaction: tax experts here, environmental law gurus here, intellectual property folks over here. Lead counsel then reviews the consolidated input and makes final adjustments.

Practice systems can usefully support the partner/associate relationship. Associates may do much of the answer configuration and drafting, while supervising partners can access features for quick review and commentary.

These systems can even be powerful tools for cooperative work among *opposing* parties to a deal or dispute. By sharing access to an interactive drafting environment, attention can be paid to high level decisions rather than specific wordsmithing, and revised documentation can be quickly generated. Positions and issues can be articulated with greater clarity.

3.6.3 Participatory knowledge modeling

Systems should invite users to participate in their growth and refinement through feedback mechanisms and associated organizational processes. They should be communally evolvable, in the sense that corrections and enhancements can easily be communicated and implemented

Some practice systems provide rudimentary forms of group authoring, by allowing annotations and textual variations to be added by users over a network. With proper incentives and management, such systems can unleash collective energies typically untapped by single-author installations. The Internet offers far greater scale for these effects, opening up such possibilities as published systems that include a built-in community of collaborating practitioners, virtual court proceedings, and online dispute resolution spaces. [8]

More advanced forms of collaboration around document *models* were explored in the Open Practice Tools initiative, which sought to apply open-source software concepts to the world of law practice automation. The basic idea was that some technologists, lawyers, librarians, knowledge managers, educators, and computer scientists would join in an open, cross-organizational, international conversation about common dimensions of their work. They would settle on some standard ways to think about and implement legal applications. Practice area by practice area, participants would evolve shared "concept maps" that identify the typical data elements, rules, and processes involved. They would contribute practice software examples and components to a common repository, while also following OPT principles in their own private or commercial applications. The anticipated result was an upsurge in legal technology innovation, productivity, quality, reusability, and interoperability. (For more about the OPT vision, see [16].)

3.6.4 Lawyer-technologist co-production

Collaboration between legal professionals and information system professionals also deserves attention. One obvious context is the development and maintenance of practice systems. Developers can weave interactive feedback mechanisms into their systems in progress, allowing users to point out errors, omissions, and opportunities for improvement right within specific application sessions. Systems can be engineered such that users can add annotations that are immediately available to fellow users. More ambitiously, lawyers can be enlisted to pseudo-code document models using standardized mark-up conventions that are semi-automatically interpretable by the document assembly engine.

3.6.5 Sharing the work

We need to think creatively about Who Does What When in the lawyer/client and technologist/lawyer relationships. Co-production offers a fertile middle ground between do-it-yourself and trust-me-I'm-the-professional-here. Tilling that soil means figuring out how tasks are best allocated for mutually effective results.

3.7 Transparency

Through current features like previews, test assembles, and answer summaries, users can ferret out some of the logic that has been programmed into the model they're using. They can usually see where their answers (and non-answers) make a difference. But they would be much better served by comprehensive tools for showing the underlying logic of models in use.

Elegant handling of unanswerdness is a critical element, both because unresolved automation provides a useful roadmap to decisions ahead (and underlying logic), and because often key facts and decisions remain indeterminate in early stages of drafting.

Today's logic-and-variable-embedded model documents ought somehow to be more intelligible to ordinary end users. Leaving the "code" or markup present and viewable in assembling documents serves to educate users about the logical structure of the model they're working with, remind them of the choices they made, and alert them to informational or decisional work remaining in a drafting project. Users should be able to see the consequences of their answers.

This involves making unresolved logic optionally visible, and even *resolved* passages with presently unselected branches shown.

Knowing what will happen in general (what questions will I be asked? what am I in for?) and in particular (what happens if I make this choice?) is important for a sense of user control. It should be easy to do quick compares of results with different answers (what if?). Given the answers I've given and the choices I've made, what options do I presently have?

3.8 Task management

In addition to form and commentary automation, systems should optionally include an intelligent checklist of things to do and things to consider. Such a dynamic queue might especially cover things to do and think about in the drafting process. "You may want to consider ..." "Things to be sure you've covered." Users should be able to mark items as done or considered (ok/dismiss/defer), and enter additional items of their own. Items should be optionally includable in drafts.

Few present document drafting systems directly support the tracking and modeling of tasks.

3.9 Historicity

Current engines don't do a great job in answering questions like "What came from where?" and "Who did what when where?"

People reasonably want to know the *provenance* of drafts, text recommendations, and commentary – where things came from – 'says who?' Users should in effect be able to ask the system to report on its actions – "Here's what I've done. You told me x; so I did y."

While the how/who/when of revisions has little bearing on the current 'what,' they can have deep significance for the pragmatics of drafting processes. Who changed what or where it came from will impact future acceptance of and attention to a particular text.

Edit histories are paths in state space, mapping transformations of texts over time. Full revision history with undo/rollback should be easy with today's cheap storage and processing.

3.10 Voice, touch, and visualization

Little has yet been done in the legal drafting context to take advantage of alternative interfaces such as voice and touch. Nor have we tapped more than the most elementary visual interfaces.

Voice mediated document assembly sessions could use voice and visual dimensions synergistically - e.g., "give me that red paragraph"; "not that blue one, the orange one"; "make all the parts that depend on this condition green."

You can imagine document components snapping together and adjusting themselves to the current logical state of the session. On a large screen-like surface that users interact with directly using their fingers.

3.11 Multiplicity and simplicity

Most of the above frontiers involve going from one to many.

- From one *rendering* of a model or draft to multiple – enabling users and authors to see texts through alternative structures and filters.
- From one *user* to multiple – enabling the whole range of roles to interact with each other synchronously and asynchronously.
- From one *phase* to multiple – enabling systems and their outputs to be used across the life of transactions, with automation and encoded know-how continuously available.
- From one kind of *artifact* (classical 'document') to many – more explicitly enabling 'assembly' of web pages, processes, checklists, and other computer-assistable aspects of legal and related work.

A fundamental challenge, however – especially in light of all the additional sophistication just described – remains how to make drafting systems and their artifacts *simple* to use and understand. Attention to human factors and usability may be the most important work yet.

4. THEORETICAL FRONTIERS

4.1 Theory in practice

Many people have thought deeply about legal document automation over the past several decades, and some built sophisticated software to embody their ideas. Their collective work represents a huge knowledge pool, albeit largely uncharted and inaccessible (except in the scattered files and sieve-like minds of people like me). Few vendors or developers have had time or interest to consider (let alone document) theoretical angles. As a result, it's not uncommon to encounter those who believe they have 'discovered' techniques that were extant twenty years ago. That produces great redundancy in document automation engine implementations, but also lots of innovation and learning.

While I've personally written many articles on practical dimensions of legal document modeling – and remain very active in real world applications – I haven't done much serious theoretical work on this subject since my Knowing Documents article at ICAIL 1993 ([14]). As a practicing legal knowledge systems architect I can only rarely justify work that steps beyond products, projects, and practical mechanics. But I've had the good fortune to interact with many deep thinkers in the document automation field, and have been privy to some spectacular visions.

4.2 Academic and research attention

Thomas Gordon [7] pointed out as early as 1989 that most commercial products lack the advantages of declarative knowledge representations, such as automated explanation, do not handle defaults and exceptions very well, and provide no support for reasoning in multiple interpretative contexts. They are still following the procedural markup paradigm associated with Sprowl, wherein master documents are expressed in terms of if-then structures, repeat loops, and variables. Neither the documents nor the legal-factual circumstances occasioning their particular configurations are explicitly modeled.

Karl Branting was among the earliest to examine the notion of self-describing documents and their potential role in new modes of expressing and delivering knowledge pertinent to legal drafting. His DocuPlanner system [6] used models of illocutionary and rhetorical structures to make goals and stylistic conventions explicit, so that documents become "queryable." The operators involved in expressing these structures constitute a grammar that can be used to generate new documents.

One neighboring strand of work has been in the legislative and regulatory drafting area. Here generally one of the goals is to produce texts that can easily be searched or automatically reasoned against, for example in question answering systems. But drafting support tools also assist in automating the construction of legal sources, for instance to ensure stylistic and content requirements and improve formal representations. Moens [20] summarizes the history and issues in this arena.

Kerrigan and Law [10] used an XML framework to introduce first order predicate calculus models into the text of regulations.

The Italian Norme in Rete project (Biagoli et al, [2]) shows how document type definitions can guide the drafting of legislation. And the new European ESTRELLA project (http://www.estrellaproject.org) has a 'workpackage' on managing legislative texts and other sources.

4.3 New theoretical directions

Many theoretical frontiers remain unsettled. Two particularly grab me.

4.3.1 Universal model

I've long felt that the legal document automation field needs a universal model of computer-aided drafting. A means for all of a system's knowledge and behavior to be explicitly declared. A conceptualization that is sufficiently rich and general to comprehend both present and foreseeable functionality. My efforts in Section 5 below are a step in that direction.

4.3.2 Darwin among the documents

How about looking for evolutionary phenomena among the cognitive tools and materials with which legal professionals work? It may be productive to apply natural-selection-among-replicators ideas to documents and precedents. The scarce resources they compete for are human attention, valuation, and use. Those with effective contents and *Baupläne* (body plans, or blueprints) live on to reproduce themselves through the preferential adoption of legal drafters. We often see "explosive speciation" of legal provisions into previously vacant niches. Provision variations can be understood as competing alleles.

Organisms conveniently "do their thing" without constant supervision. Could we somehow "grow" drafting systems? Will it eventually make sense to talk about domestication, cultivation, hybridization, breeding, and genetic engineering in relation to our practice tools? How do we inject natural hardiness into our artificial systems?

5. FRAGMENTS OF GENERAL THEORY

Legal work is a dance of knowledge, deliberation, and action. One of the most common and important kinds of action in that dance is text preparation. Documents play into most of what lawyers know and do. One of the most important forms of legal technology therefore is automated drafting support.

People in my line of work build machines with words. Documents that emit documents. Texts that act. Here are fragments of a general theory of such machines.

5.1 Getting ready

An adequately general theory of legal document automation is assisted by adopting four preliminary attitudes:

- step back from the mechanics of specific software,
- focus on the knowledge being expressed,
- think beyond the documents, and
- pretend we are not constrained by computing resources.

We'll do best to work backwards from an imagined ideal world of perfect memory, with unlimited storage and processing power, and ignore computational efficiency and other practicalities for the moment. Let's also assume complete user freedom to say whatever they want about anything. All of these assumptions can later be constrained as necessary.

Most of my examples here relate to drafting of the sort that occurs in a law office in connection with business transactions, but the principles are generalizable to other contexts. The goal is to formulate a conceptual framework that works for *any* kind of text composition.

5.2 Basic distinctions

Looseness of terminology (like 'logic' and 'field') and conflation of differences (like among different hierarchies and non-hierarchies – composition, containment, condition nesting, kind-of, version-of) have caused considerable confusion. Some of that confusion can be dispelled by recognizing the fundamental distinctions between:

- In and About – what is said in/by a text vs. what is said about a text

- Is and Ought – what is (or isn't) in a text vs. what should/shouldn't be in it
- Which and Where – what should be included vs. where it should it appear

Current word processing and document assembly tools make it different to express those differences. Microsoft Word, for instance, typically collapses text and metatext (e.g., comments) into a single artifact. HotDocs and other assembly engines don't let you easily express that something should occur *somewhere* in a document, without a relative location being specified.

5.3 Drafting systems

5.3.1 Drafting system

A drafting system consists of

- one or more people engaged in the composition of texts,
- one or more structured collections of texts (textbases),
- software processes operating on those textbases (autonomously or at the direct behest of people), and
- the physical devices needed to store, process and interact with the texts.

5.3.2 Functions

Drafting typically involves working *on* texts *with* texts.

A legal drafting system is designed to help people create texts. It does that by (1) enabling them to search for, browse, read, and copy existing materials and guidance, (2) accepting new texts, some of which say things about existing texts, and (3) generating texts of various sorts that change as users interact with them.

5.3.3 Knowledge components

A wide range of knowledge components (often encountered in conventional forms and commentaries) are found in drafting systems. For example:

- What issues need to be dealt with and considerations on their various resolutions
- Who knows about a kind of transaction
- What words best go where when
- Variations to consider (aggressive stances, compromises)
- What to expect from other parties
- How to react to other side's rejection of certain provisions
- How to gather needed information and raw material
- Questions to ask the client
- Stylistic rules and conventions (e.g., 'house style')

Some forms of knowledge aren't well communicated in flat, static documents. Dynamic, or modeled, content – with variability, conditionality, repetition, and annotation – serves much better.

5.3.4 Commentaries

Many forms of commentary are involved. They may be

- about a particular kind of transaction and its forms in general, or
- specific to a given matter.

They come in different *kinds*, such as:

- explanation (of history, purpose)
- guidance (suggestion, recommendation)
- notes about alternative wordings
- links to auxiliary resources

and have different *subjects*, such as

- a transaction as a whole
- a particular form as a whole
- a specific issue or decision (transactional or drafting)
- a particular passage or context

They can be *accessed* in different places and ways. Some naturally are associated with locations in the form, some with questions to be answered, some with both. Commentary text may need to vary based on whether it is in context or in standalone compilation. They may be *stored* in the application or in external sources such as websites or databases. Resources may include links to other memos in a firm's file system, intranet or web pages, or to experts who can be contacted.

5.3.5 Tasks

There are also many different kinds of things to do, consider, and decide:

- those unconditionally suggested in any transaction of the sort covered
- those conditionally suggested based on deal characteristics
- those entered manually by user (specific to a transaction)
- intra-documentary (specific to a particular document)
- inter-documentary (relating to more than one)
- extra-documentary (pertaining to issues and actions outside of any document)

5.3.6 Triggers for variation

There are many different triggers for variation in a drafting system, any of which may impact model language, associated commentary, or suggested issues/actions. For example:

- client type and identity
- counterparty type and identity
- counterparty counsel identity
- industry
- transactional terms
- transactional *events* (such as a position taken by a counterparty during negotiation)
- user preferences

5.4 Texts and metatexts

Legal practice systems are usefully conceived as richly interwoven fabrics of texts and metatexts. This textbase is a network of objects, a dynamic collection whose nodes and links change as users interact with it.

Three basic kinds of texts are involved:

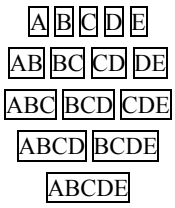
- texts intended for some purpose outside the drafting system (including those presently in progress), which I will refer to as "documents";
- textual models and commentaries of various sorts, which I will refer to as "metatexts," because they are *about* other texts; and
- texts that are neither documents nor metatexts, but play a role in drafting work, such as scripts, plans, plan models, and interface definitions.

Let's consider the associated objects in the imagined network.

5.4.1 Text

A **text** is a consecutive set of characters or other graphemes (visual units of potential linguistic significance.)[2]

Most texts consists of many sub-texts. For instance, the text 'ABCDE' includes the following fifteen texts, any of which might be the subject of attention:



These can also be understood as falling into this natural hierarchy, expressed as an outline (omitting duplicate nodes):

ABCDE
- ABCD
  - ABC
    - AB
      - A
      - B
    - BC
      - C
  - BCD
    - CD
      - D
- BCDE
  - CDE
    - DE
      - E

Since the number of contiguous sub-sets of a text of N characters is $(N^2 + N)/2$, all non-trivial texts have quite a few. For instance, a text of 100 characters has 5,050. A text of 10,000 words averaging five letters, together with an equal number of spaces and punctuation, has $(60{,}000^2 + 60{,}000)/2$ or 1,800,030,000 sub-texts. For each character added to a text of length N, N+1 new subtexts become specifiable.

Of course the vast majority of sub-texts are of no interest whatsoever. But optimal text editing and modeling require that any arbitrary sub-set be addressable, down to the individual character or glyph. (The inability of most commercial legal document management systems to address texts below the document level, or some document assembly systems to address objects below the paragraph level, has posed a major limit on expressiveness.)

5.4.2 Metatext

A **metatext** is any text that makes a statement about another text, within or across text objects. A text can also say something about itself, or about another metatext. It can be about a *specific* text, or a defined class of texts.

The things that can be said about text fall into many categories:

- topic (what the referred-to text is about)
- source (where it came from)
- appearance or formatting (how text segments should be formatted or laid out, e.g., in bold, or in a footnote)
- structural (what part of a part-of hierarchy a segment plays, e.g., article, section)
- semantic (e.g., date of agreement, plaintiff name)
- purposive (why or teleology; legal or business requirements)
- explanatory (manual or machine built annotations)

Some kinds of metatexts are peculiarly at home in intelligent drafting systems. These include:

- logical (variable fields; conditional or repeating text)
- text models (descriptions of required and permitted elements, e.g., using XML DTD or schema syntax)

Some kinds of texts and metatexts are more pertinent and useful for drafting purposes than others, but there's no bright dividing line between those that are and aren't.

5.4.3 Text objects

Text objects – the nodes of the imagined network – include documents, files, email messages, and other kinds of records or containers used to store, identify, or locate texts. The granularity of reification is an engineering decision. Specific features of the 'about' links below enable references at sub-object levels so that arbitrary precision can be achieved.

5.4.4 Versions

A version of a text comes into existence when it is copied, and the copy can independently be edited. You can also imagine each change in a document under edit to result in a separate virtual version.

Different *renditions* of a document likewise are different versions. An italicized character is a different character than a non-italicized one. Even font differences can have intended significance. Each state of a text that involves any difference in its contents is a potentially distinct version.

---

[2] For present purposes I will mostly talk in terms of texts that can faithfully be captured in linear (one-dimensional) sequences. Of course many documents leverage two-dimensional layouts to express meaning, and such higher-dimensional orderings need to be taken into account in a full theory.

All texts, including metatexts, can be versioned. Deciding which metatexts of a given text can appropriately be replicated for a new version of that text raises both accounting complexities (computing the correct address of the referent text in the revised object) and semantic challenges (when does a tiny text edit render the prior commentary invalid?)

5.5 Links among texts

5.5.1 About link

'About links' connect two text objects – or locations or segments within them – where the first is about the second. You can think of them as directed links or arcs in a network diagram, with text objects or ranges being the nodes. They memorialize aboutness.

Sometimes texts are marked up with special tags to signify the locations or passages about which something is being said, but such tags are not really "in" the text itself. They signify the intended object of some *other* text, a metatext, while collapsing both into a single virtual layer. Markup schemes like SGML and XML of course are quite useful insofar as they are both machine and humanly readable. [21], [22].

Figure 1 shows the several basic ways one text can refer to another. The bottom square box represents a text object, with a series of characters and in-line text objects. The leftmost text object above has something to say about the bottom object as a whole – for instance, "This is a contract." The second object says something about a particular inter-character location – for instance, "Additional description of performance required here." The third object refers to a single character – for instance, "Underline this." The fourth object says something about a range of characters and objects – for instance, "Include only if counterparty is Stanford Law School." You could additionally have an object that refers to a discontiguous collection of locations and/or passages within another text object.

Figure 2 shows how sub-texts of one text object might be linked to sub-texts of another object. (There's also an example of a link from one part of an object to another part of the same object.) Figure 3 shows a simple example with multiple texts and metatexts.

About links have the following properties:

- the address of the metatext (where the content can be found)
- the specified subject(s) (text object, location, or characteristics of referent)
- the kind of statement being made (topic labeling, source describing, logical modeling, etc.)
- the identity of the person or process asserting the link (Says who?)
- the date and time of assertion

5.5.2 Version link

Directed links can be drawn from each text to objects that have started out as copies of it. Is-a-version-of links can also be drawn from *part* of one text to part of another when the first has been copied into the second.

When someone adds or deletes a character, every supertext containing the location at which that addition or deletion occurs is changed. But only such texts as are affirmatively saved as discrete objects constitute versions.

A single word processing operation like sorting a list or searching and replacing can have many local changes that are unimportant compared to the 'macro' change. It makes sense to store the details of such operations as part of the metadata of the link between versions, so that the forest is not lost in the trees. (It's also a much more compact and meaningful representation of the delta.)

5.5.3 Similarity links

A system or its users can notice texts that are identical or highly similar to each other, and connect them with undirected similarity links. Versions of a text naturally lend themselves to similarity links.

5.5.4 The Textbase as network

The entire web of texts and links that can be accessed by anyone within a drafting community is a unified network of text objects connected by links, mostly directional, and all with types and other attributes. All nodes are texts, but all texts are not necessarily nodes. (They may be unindividuated sub-texts.)

A real network will of course be vastly more complex than Figures 3. There may be millions of text objects, many with multiple versions, and millions of links connecting versions of both texts and metatexts. Intelligent filters are needed to compose relevant subsets for drafting sessions.

5.6 The actors and actions

Two kinds of actors interact with the above text network – people and software agents. It's useful to remind ourselves what kinds of things they do.

5.6.1 People

Document process verbs most associated with human actors include:

- finding examples and other raw material (resources, grist, fodder – both in-house and in the outside world), such as
  - similar transactions
  - similar whole documents and sets
  - similar components, clauses
  - people with relevant experience
- stitching pieces into new draft
- replacing old transaction-specific material with analogous material for new transaction
- composing fresh texts
- commenting, critiquing
- negotiating - proposing, reacting, arguing
- revising
- comparing
- proofreading, checking
- finalizing, "freezing"
- modeling
- managing

Note the knowledge tasks going on:

- knowing/deciding *what* to say
- knowing/deciding *how* and *where* to say it
- knowing what one needs to know and decide the above things

5.6.2 Software

Software performs a wide range of tasks in a drafting system:

- elicit and accept input
- present or display texts and text-like interfaces
- inform
- educate
- remind
- suggest, advise
- supply links to related resources
- warn, alert
- render
- format
- assemble
- record
- learn
- notice gaps, conflicts, ambiguities in models
- interpret, translate

Software thus can serve as an observer and text maker/editor – a speech-actor in its own right. It acts as a model processor, actualizer, implementer, enforcer, manager, executor, and intelligent assistant.

There can be different levels of reaction to human edits:

- watching/observing/recording vs. intervening
- changing something in the current document or interface (maintaining model compliance or semantic consistency)
- permitting/forbidding/preventing the change
- suggesting changes - volunteering possibilities
- analyzing networks of links
- alerting, reminding

5.7 Document modeling and drafting with networks of texts and metatexts

Modeling and drafting legal texts within systems based on networks of texts and metatexts would be substantially different from today's practices. And I believe radically more powerful. Here are some of the differences, and aspects of how I imagine this working.

- An architecture that separates texts about texts from the text they are about enables *anyone to say anything about anything at any time*, without concerns about file locking and collisions. Commentary and modeling become fully distributed activities. They can occur as incrementally, incompletely, and informally as people like. Many voices can be heard, and they need not all agree. Drafters can use as much or as little of that as they wish. Yet both people and software agents can draw such materials into operative models when appropriate, with full traceability back to their origins.
- Modeling can be expressed with respect to entire community repositories, rather than within the confines of discrete models or templates. You might still create discrete models, with associated clouds of metatexts, but those metatexts will be straightforwardly available for use in other models and less formally modeled contexts.
- You can express ideas and opinions about texts that are neither location- nor instance- specific. "Whenever you're drafting an XYZ agreement, be sure to cover topics A and B *somewhere*."
- Taxonomies can be bottom-up and folksonomic – and thus emergent and resilient, rather than top-down and brittle. Anyone can label anything anyway they wish, although they are increasingly guided by the patterns that emerge.
- Drafting sessions can be guided by dynamically assembled collections of modeling metatexts, filtered as needed by user preferences and harmonized as necessary by automated routines.
- Specific moves in a system-user drafting session can be modeled as standardized state transitions from prescriptive to descriptive metatext. For instance, when a user has adopted and chosen to process a passage containing a location as to which a field metatext has been associated ("Insert plaintiff name here"), her interaction with the system would result in the given answer ("Smith") being tagged with the metatext "Name of plaintiff." Similarly, a location starting out with an associated metatext that instructs one to include some passage IF a certain situation obtains becomes a passage that is the referent of a metatext explaining that it is there BECAUSE User X said that that situation obtained.
- As drafters work they can consult dynamically assembled windows of metatexts that are associated directly or indirectly with the passages of text in draft that are currently in focus.
- Texts that have been composed within a drafting system of this sort will be far more richly described and thus more automatically re-processable.
- The network itself can be continually mined for collective knowledge, using techniques like the PageRank algorithm for instance to compute the 'goodness' of particular precedents and metatexts.

In short, I believe that a drafting system based on operations against a network of texts and metatexts offers promising developments on all of the frontiers described in Section 3 above.

5.8 Engineering challenges

There are daunting challenges to be faced in achieving the benefits of this new paradigm. The continuous recomputation of text networks needed to deliver satisfactory performance for a single law office could well require computing resources on the order of those now deployed by Google for mapping the entire global web. Ten years from now, that won't seem like such a tall order.

Here are some of the engineering challenges:

- representing texts and links compactly
- efficiently processing the sparse matrices implicit in the network architecture
- merging textbases from two organizations
- faithfully maintaining links when text is rearranged
- delivering smart cut & paste tools (e.g., to reconcile models, or to intervene before paste is committed)
- reconciling inconsistencies between models defined or implied by sets of metatexts

5.9 A further step - Pantextualism

Other components of a more comprehensive *practice* system can effectively and intuitively be expressed as texts. While these are typically not freestanding documents themselves, they can be referenced *in* such texts. And while they are not metatexts (because they are about nondocumentary things), there of course can be metatexts about *them*.

5.9.1 Matters and projects

Pieces of information about the case, matter, transaction, or other project being worked on can be straightforwardly expressed in text-like data structures that consist of attribute/value (or question/answer) pairs. Metatexts can be used to prescribe and describe such structures.

5.9.2 Histories and plans

Sequences of actions that have taken place – or could/should take place – are naturally expressed as texts. Histories or narratives contain ordered lists of what is claimed by someone to have happened. Task (or to-do) lists describe what may or should/shouldn't happen. Only future-oriented plans of course are appropriately marked up with variables, conditions, and open iterations.

5.9.3 Interface

The interface through which the user operates (both its appearance, e.g., HTML, and its behavior, e.g., JavaScript) is essentially just another kind of text, one which can be dynamically assembled based on models and session data.

5.9.4 Code

Scripts and programs used to control other aspects of system behavior of course are also texts – which can be modeled and assembled.

Modeling all of the knowledge objects in a drafting system as texts and links among texts provides a conceptually satisfying strategy with tantalizing engineering possibilities. I hope to refine, expand, and formalize these ideas in a later article.

6. FUTURE DRAFTING

Despite the vast amount of practical and theoretical work that has been done in the legal document automation field, we've barely dented the surface of opportunity. Only a small fraction of appropriate applications has been deployed, and significant improvements in products and theory remain unrealized.

Application progress of course is mostly driven by what markets want (or at least what vendors *think* markets want, and think can be delivered profitably). Not only is demand unclear for advanced features such as those reviewed here, but significant investments are required to deliver them commercially. Still, those features will enable much more powerful and satisfying drafting experiences, and will draw new attention from previously reticent drafters. While the transition will likely be gradual, waves of new functionality could result in pervasively adopted drafting methodologies largely unseen today:

- When you have a drafting project that doesn't involve a radically new or unusual document, you will typically be guided by a rich fabric of modeling texts, drawn either from an in-house collection, or from a published set.
- You'll interact with that modeled knowledge via intelligent drafting tools that let you work simultaneously in an interview-like outline of decisions and inputs, and an evolving, dynamic draft. Both the interview and draft will be configurable, filterable, and easily navigable.
- You'll be able to access and add commentaries and resources in both modes, and turn on views that reveal alternative details of both your draft and the web of texts to which it relates.
- Even for model-less contexts, text-network-savvy drafting tools will be commonplace.

When all is said and done, legal drafting is about what can be said and done with texts. Our tools should allow us to weave and navigate richer networks of text as we prepare documents to accomplish our legal purposes.

ACKNOWLEDGEMENTS

My thanks to Craig Kobayashi, Darryl Mountain, Lavern Pritchard, Mark Larson, Tom Gordon, and several anonymous reviewers for useful comments on prior drafts of this article. This remains a work in progress and further input is heartily welcomed.

REFERENCES

[1] Armstrong, S. and Lauritsen, M. Working Smarter to Help Lawyers Work Smart: Linking Education and Information Technology. *Law Firm Governance*, Summer 1999.

[2] Biagioli, C., Francesconi, E., Spinosa, P., and Taddie, M. A legal drafting environment based on formal and semantic XML standards. In *Proceedings of the Tenth International Conference on Artificial Intelligence and Law*. Bologna, June 2005. pp. 244-245

[3] Branting, L. K. Techniques for Automated Judicial Document Drafting, *International Journal of Law & Information Technology*, 6(2):214-229 (1998).

[4] Branting, L.K., Lester, J., and Callaway, C. Automating Judicial Document Drafting: A Discourse-Based Approach, *Artificial Intelligence and Law*, 6(2-4):105-110 (1998)

[5] Branting, L.K., Callaway, C., Mott, B., and Lester, J. Integrating Discourse and Domain Knowledge for Document Drafting. Oslo, Norway, July 14-17, 1999

[6] Branting, L.K., Lester, J., and Callaway, C. A Framework for Self-Explaining Legal Documents. *Proceedings of the Sixth International Conference on Artificial Intelligence*

and Law, June 30-July 3, 1997, University of Melbourne, Melbourne, Australia, pp. 72-81

[7] Gordon, T. 1989. A Theory Construction Approach to Legal Document Assembly. In *Pre-Proceedings of The Third International Conference on Logic, Informatics, and Law*, 2:485-498. Florence.

[8] Greenspun, P., and Lauritsen, M. Making Way for Intelligence in Case Space. In *Proceedings of the Fifth International Conference on Artificial Intelligence and Law*. College Park, Maryland, June 1995.

[9] Hokkanen, J. and Lauritsen, M. Knowledge Tools for Legal Knowledge Tool Makers. *Artificial Intelligence and Law*. (2003)

[10] Kerrigan, S. and Law, Kincho. Logic-Based Regulation Compliance-Assistance. In *Proceedings of the Ninth International Conference on Artificial Intelligence and Law*. Edinburgh, June 2003, pp. 126-135.

[11] Kiefer, D. and Lauritsen, M. Recent Developments in Automating Legal Documents. 52 *Syracuse Law Review* 1091 (2002)

[12] Lauritsen, M. and Janis, B. Going the Last Mile. *E-filing Report*, December 2004.

[13] Lauritsen, M. Technology Report: Building Legal Practice Systems with Today's Commercial Authoring Tools. 1 *Artificial Intelligence and Law* 87-102 (1992)

[14] Lauritsen, M. Knowing Documents. In *Proceedings of the Fourth International Conference on Artificial Intelligence and Law*. Amsterdam, June 1993.

[15] Lauritsen, M. A Dispatch from the Document Automation Trenches. Workshop on Automated Document Drafting. Seventh International Conference on Artificial Intelligence and Law. Oslo, June 1999

[16] Lauritsen, M. Ontologies and Openness in Law Practice Automation. Workshop on "Legal Knowledge Systems in Action," Eighth International Conference on Artificial Intelligence and Law, St. Louis, Missouri. May 2001 [http://www.capstonepractice.com/OntoOpen.html]

[17] Lauritsen, M. Fall in Line with Document Assembly, *Law Office Computing*, February/March 2006, pp. 66-75

[18] Lauritsen, M. Artificial Intelligence and the Real Legal Workplace. In Lodder and Oskamp eds., *Information Technology & Lawyers: Advanced technology in the legal domain, from challenges to daily routines* (Springer, 2006), pp. 165-176

[19] Lauritsen, M. and Soudakoff, A. Shopper's Guide to Legal Document Assembly. *Law Office Computing*, October/November 1997

[20] Moens, M. Improving Access to Legal Information: How Drafting Systems Help. In Lodder and Oskamp eds., *Information Technology & Lawyers: Advanced technology in the legal domain, from challenges to daily routines* (Springer, 2006), pp. 119-136

[21] Poulin, D., Huard, G, and Lavoie, A. The other formalization of Law: SGML modeling and tagging. *Proceedings of the Sixth International Conference on Artificial Intelligence and Law*, June 30-July 3, 1997, University of Melbourne, Melbourne, Australia, pp. 82-88

[22] Rothman, A. and Lauritsen, M. Expect XML to Change the Practice--Markedly, *National Law Journal*, February 1, 1999, p. C1.

[23] Sprowl, J. 1979. Automating the Legal Reasoning Process: A Computer that uses Regulations and Statutes to Draft Legal Documents. 1 Am. B. Found. Res. J. 1-81

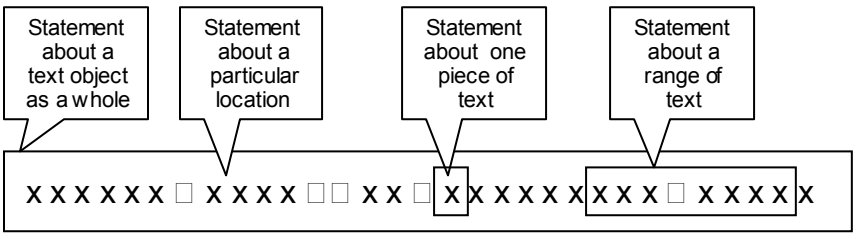


Figure 1

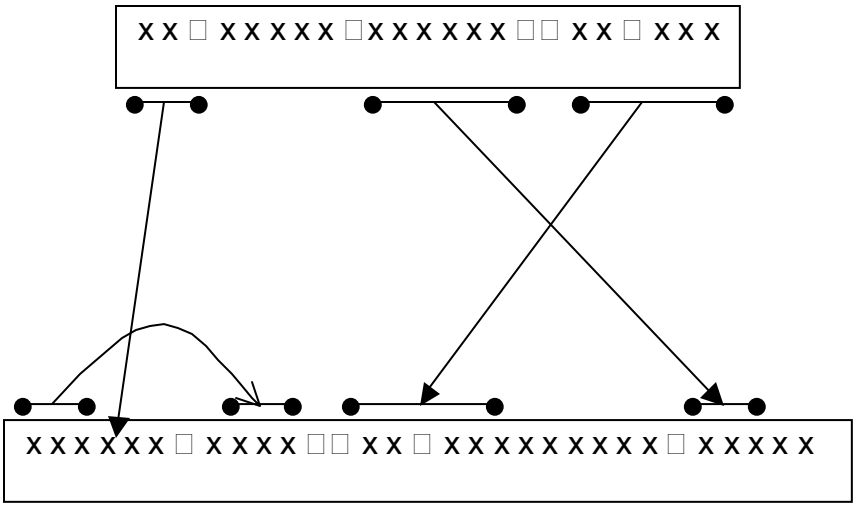


Figure 2

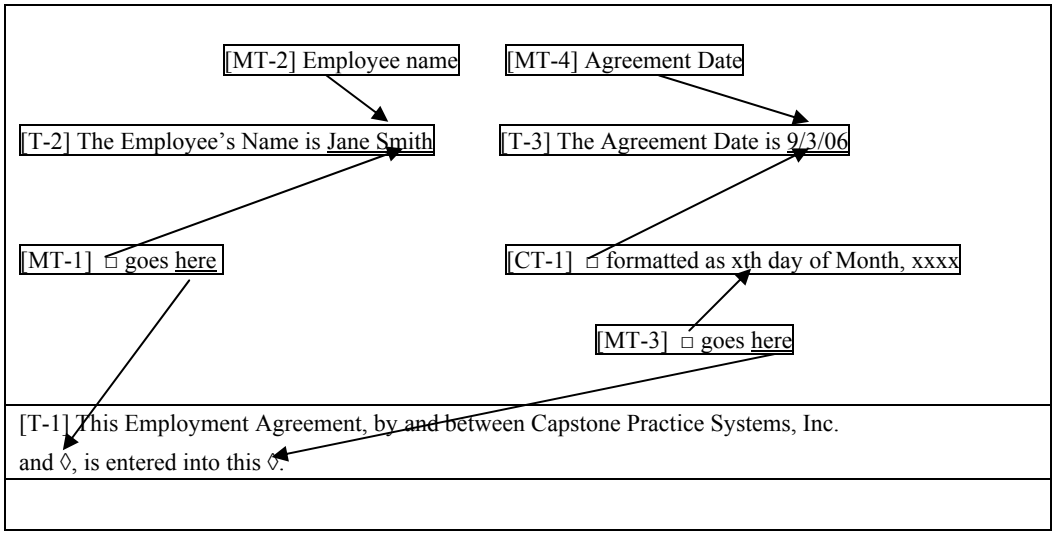


Figure 3