Building Interactive, Data-Gathering Forms for BPM-Defined Workflows

BPM suites provide functionality for building data-gathering forms within workflows. But some workflows, especially those that involve the generation of complex legal documents, require data-gathering forms that are more complex than what virtually any BPM suite can handle.

This disconnect between form-building functionality in BPM suites and the needs of workflow users (to generate sophisticated, transaction-ready documents within the workflow) results from the data-first approach taken by BPM vendors. The data-first approach (which, for the purpose of this paper, is defined as information gathered for some reason other than document automation and then repurposed for document automation) is effective for any number of workflow scenarios.

Where the data-first approach falls short, though, is in workflows having sub-processes for generating sophisticated legal documents—contracts, for example—which can possibly require hundreds of discrete data items in addition to what an enterprise might have stored in its existing data records. Beyond the sheer volume of document-specific information that may need to be gathered is the inherent, rule-based structure of the documents themselves, which structure dictates when and if discreet data items need to be gathered at all.

To accurately gather all the information necessary for a document, to prescribe the conditions under which the information gets gathered, and to ultimately use the information to generate a transaction-ready document (or set of documents), workflow architects need to take a different approach—a *document-first* approach. The document-first approach, as the name implies, requires that a workflow architect build business rules into the document(s) before beginning the process of building data-gathering forms.

The document-first approach requires that an additional class of technology be employed—document automation process software, which not only allows for the modeling of the document(s) (building business logic into the document(s)) but enables interactive form design based on the structure and data requirements of the documents in question. Enterprise-grade document automation platforms provide APIs for BPM and other web application integrations.

Table of Contents

The Document-First Approach

The process of building data-gathering forms for the purpose of automating documents begins with the documents themselves. An architect must first build business logic into a document, a process commonly referred to as document modeling. Only after the document is fully modeled (automated) will the architect know what data needs to be gathered and under what conditions.

An architect must first build business logic into a document, a process commonly referred to as document automation. Only after the document is fully automated will the architect know what data needs to be gathered and under what conditions.

The Rule-Based Structure of Legal Documents

Most types of legal documents have an inherent structure: a combination of boilerplate text, conditional text, repeating text, and simple/computed variables. Take, for example, the sales contract represented in Figure 1.

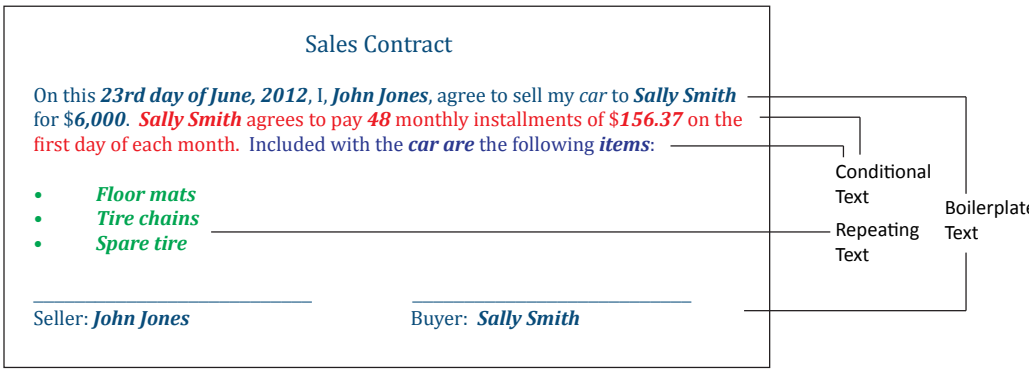

Figure 1. Simple sales contract with color-coding to demonstrate document structure.

The contract, which is obviously not meant for actual use, is very simple and represents only a fraction of the types of structure that might exist in an actual legal instrument. Even still, the document is highly structured, as is depicted by the color coding:

- Blue—boilerplate text (included in the document under all scenarios).
- Red—conditional text (included in the document only if the parties agree to a structured payout)
- Purple—conditional text included only if additional items are involved.
- Green—repeating text nested inside conditional text (The repeating text is only included if the purple text above it is included.)

Note also that italicized text in the example document represents simple variables (non-computed variables) and that the underlined text represents computed variables, of which there are three: (1) the amount of the monthly installment payment (the result of adding the simple interest to the purchase price and dividing by the number of months of the payout), (2) the word “are” (the plural form of “is”) and (3) the word “items” (a plural form of “item”). Both of these plural words, of course, would become singular if only one item were included with the sale.

An automated version of the simple sales contract is represented in Figure 2.

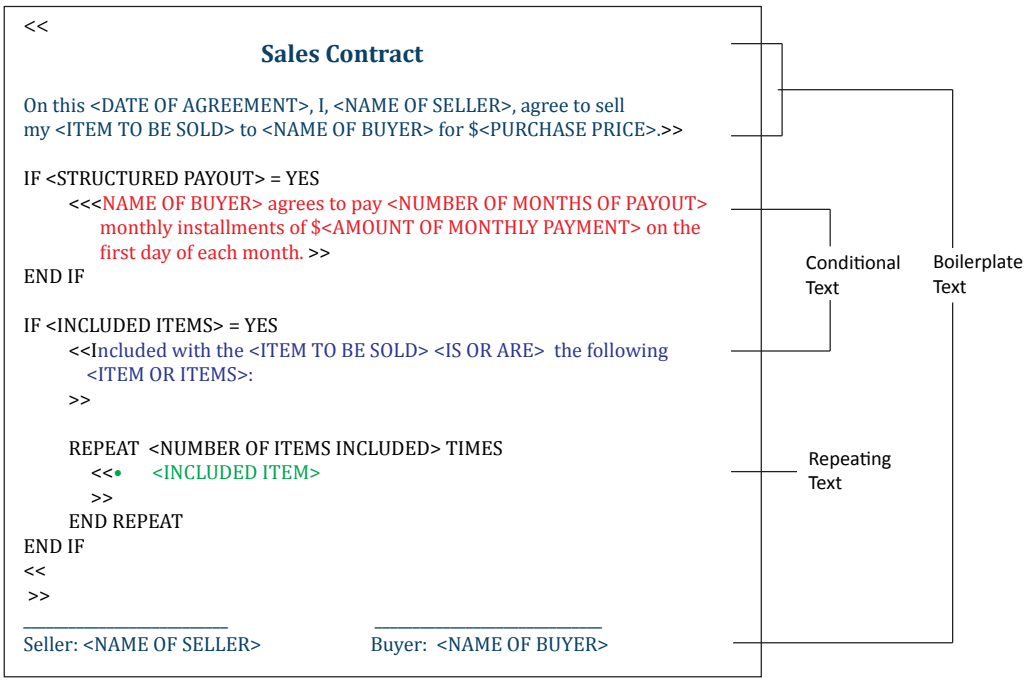

Figure 2. Business logic built into the simple sales contract.

With the business rules applied, the document's inherent structure is even more obvious. Conditional text is nested inside IF statements. The repeating text (the list of items included with the sale) is nested inside a REPEAT statement, which, itself, is nested inside an IF statement. Boilerplate text is not nested inside an IF statement, which means it will be included in the document under all conditions.

It is important to note that several variables are merged into boilerplate text and conditional text. With virtually all document automation platforms, variables used in the document become questions in the forms.

It's this concept—asking only the relevant questions—that creates the inexorable link between document structure and forms structure, which structure must reflect the scripting logic built into the documents.

The Reflexive Structure of Information-Gathering Forms

The underlying premise behind forms designed for document automation is to ask only the questions that are necessary to assemble the document(s). In other words, given all the conditional scripting in complex documents, many questions (perhaps dozens, or even hundreds) would be irrelevant to a particular matter, given an existing set of conditions. Presenting such questions in the forms would be distracting, time consuming, and potentially misleading to workflow users. It's this concept—asking only the relevant questions—that creates the inexorable link between document structure and forms structure, which structure must reflect the scripting logic built into the documents.

Boilerplate Text = Boilerplate Questions

Boilerplate text is always included in a document. Consequently, variables merged into boilerplate text must be displayed as questions in a form under all circumstances. For example, in the example sales contract, the boilerplate text includes several

simple variables. Because all of these variables are merged into boilerplate text, they could be grouped together in the same form, which would be displayed under all circumstances each time the document was generated (See Figure 3).

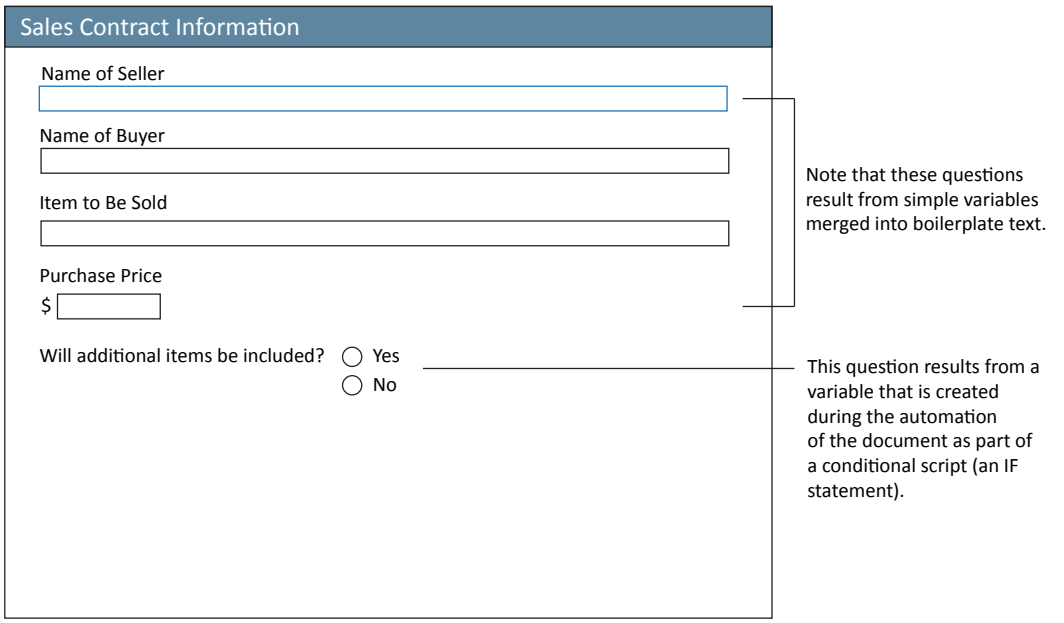

Figure 3. Form with questions from boilerplate text.

Variables merged into conditional text must be displayed as questions in a form only if the appropriate conditions exist.

Conditional Text = Conditional Questions

Variables merged into conditional text must be displayed as questions in a form only if the appropriate conditions exist. For instance, in the example sales contract within the structured payout clause, a simple variable is merged (<NUMBER OF MONTHS OF PAYOUT>). A question for this variable would only be asked in a form if the answer to a previous question were yes (See Figure 4).

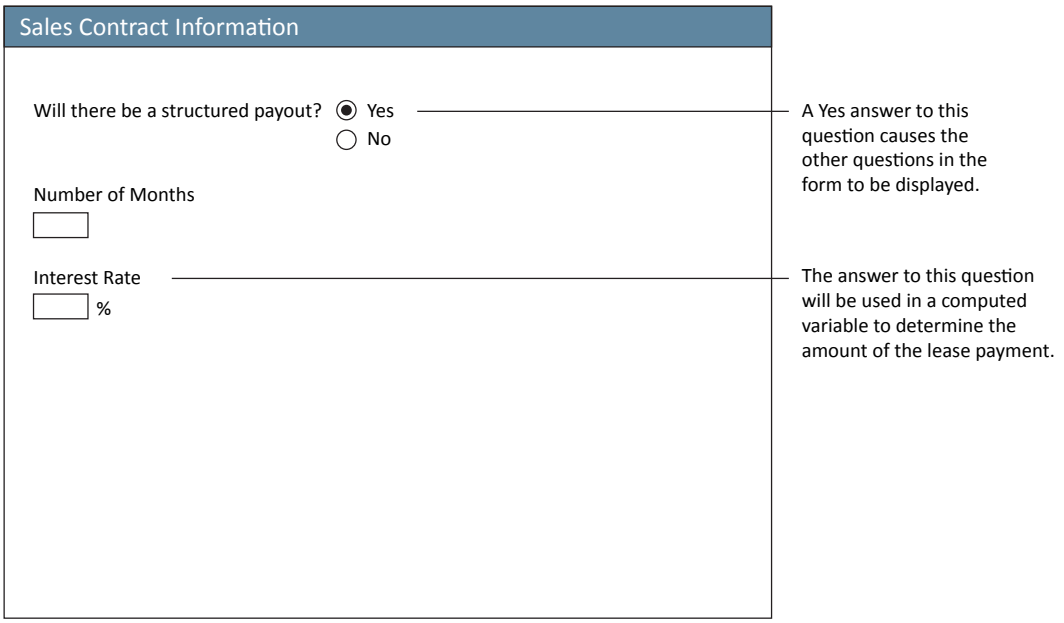

Figure 4. Interactive Form

Repeating Text

Variables merged into text within a REPEAT instruction are, themselves, repeating variables. Given the way the business logic is written into the example sales contract, a question must be asked that will determine how many fields will be displayed in the form for the included items (See Figure 5).

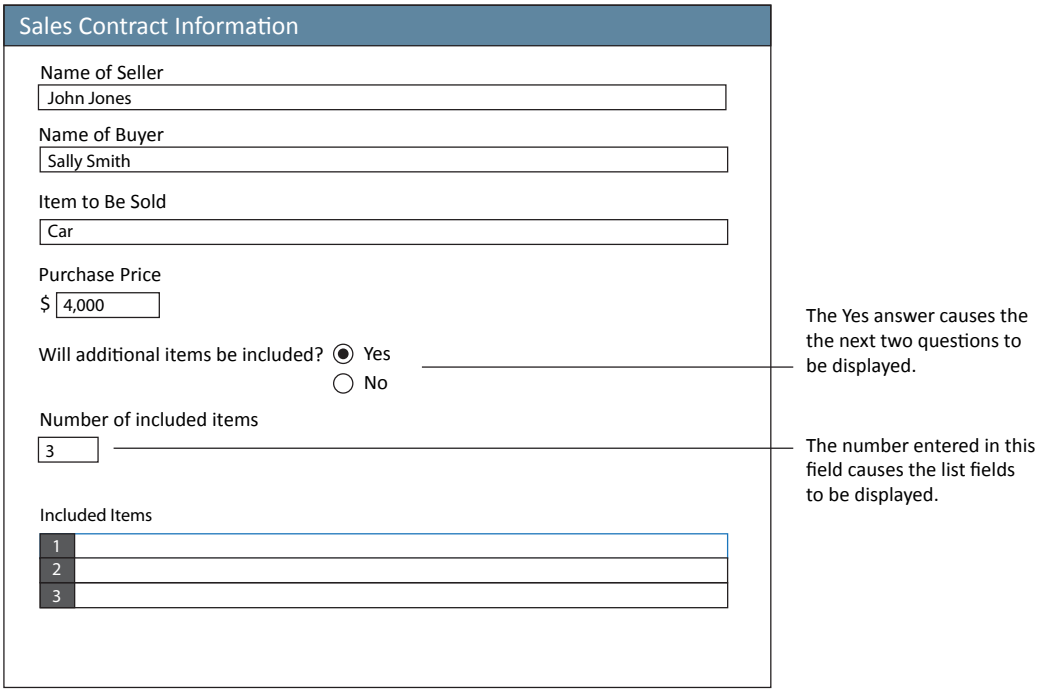

Figure 5. Form with repeating text.

A well designed interview should go beyond merely asking questions; it should actually guide and assist workflow users in getting the answer to each question entered correctly.

Note, again, that the repeating text in the simple sales contract is, itself, nested inside an IF statement, meaning that the repeating text will only be included if a particular condition exists (IF <INCLUDED ITEMS> = YES).

While the example sales contract is extremely simple, it nonetheless demonstrates the two critical points—that documents have a rule-based structure and that document structure must be reflected in forms structure so as to ask only the relevant questions based on existing conditions.

Advanced Forms for Document Automation Workflows

Because of the nature of many types of legal documents—rule-based, large data-sets, complex—a number of information-gathering forms are often required for a document or set of documents. Such a sequence of forms is commonly referred to as an interview. A well designed interview should go beyond merely asking questions; it should actually guide and assist workflow users in getting the answer to each question entered correctly. A few of the features that enforce answer quality are as follows:

- Range validation
- Resources
- Outline view
- Required answers

- Answer selection
- Interactive Forms
- Conditional Forms

Range Validation

Range validation is a method by which an architect can ensure that an answer falls within an acceptable range. For example, the acceptable term for a structured payout could be set at between 12 and 48 months, depending on negotiated terms. A validated question will not accept an answer that falls outside the prescribed range. Range validation is possible for any numeric or date variable and is frequently used to eliminate the possibility of legal exposure resulting from a typographical error or otherwise wrong answer.

Resources

A resource is a help screen that can be attached to any specific question. A resource could include an explanation of how to answer the question correctly, hyperlinks to outside resources, or technical aids, such as a spreadsheet or calculator to assist the workflow user in arriving at the correct answer.

In a document-centric workflow that includes the generation of complex documents, the easiest approach is to supplant BPM forms with an interview built to reflect the structure and business logic of the document.

Outline View

Within the context of an interview consisting of dozens of simple, interactive, and conditional forms, it's extremely helpful, if not critical, that workflow users be able to envision an outline of all of the forms in an interview. This functionality is commonly referred to as Outline View and should be an option in enterprise-grade document automation platforms. Among other things, Outline View enables workflow users to easily move backward and forward among the forms in an interview.

Required Answers

Within the context of a particular form, a template architect could designate that a question must be answered before moving on to the next question.

Answer Selection

For variables requiring an answer that falls within a predefined list (for example, counties within a state), a dropdown list can be created for the variable. In other words, rather than typing in the answer, the user would simply select it from a list.

Interactive Forms

An interactive form includes both boilerplate questions (created from variables merged into boilerplate text) and conditional questions (created from variables used in scripting instructions or merged into conditional text) (See Figure 4). Interactivity in the form results from the answer to a question causing other questions to be displayed within the form.

Conditional Forms

Conditional forms are entire forms of questions that are displayed only if certain conditions exist. Such forms would be created in situations where a document included large blocks of conditional text, which itself included many items of data.

Conditional forms, of course, can—and often are—interactive, meaning that answers provided to some of the questions in the form cause other questions to appear in the form.

Integrating Document Automation Interviews with a Workflow

In a document workflow that includes the generation of complex documents, the easiest approach is to supplant BPM forms with a document automation interview built to reflect the structure and business logic of the document. Data already on hand—stored in an existing database—can be passed into the workflow and can pre-populate fields in forms built in the document automation system. The remainder of the questions can be answered by the appropriate parties who have access and rights within the workflow.

A completed interview can be routed for approval prior to the generation of documents, and then the documents can be passed back into the workflow for routing and approval.

. . . For enterprises that produce complex legal documentation and require transaction-ready instruments out of the workflow without after-the-fact editing, an enterprise-grade document automation system is critical.

Summary

It should be noted that not all document automation systems emphasise sophisticated interview design and construction. Many systems focus, instead, on ease of use, an approach that is especially applicable for organisations that have just a few simplistic documents or for whom the objective is generating a good first draft.

However, for enterprises that produce complex legal documentation and require transaction-ready instruments generated within the workflow that require no *after-the fact* editing, an enterprise-grade document automation system is critical. Such a system should allow for any level of document modeling, no matter how complex the documents may be, and should facilitate reflexive interview design and development. Finally, an enterprise-grade system will allow for deployment in multiple environments and should include highly evolved APIs for the purpose of workflow and other system integration.

HotDocs Corporation

387 S. 520 W.
Suite 210
Lindon, Utah 84042

sales@hotdocs.com

(801) 615-2200

HOTdocs™
automated document assembly