XML Conference Schema Documentation

Lisa **Richards**

Abstract

An important part (indeed the core) of the business of legislatures or parliaments is drafting, scrutinising, and changing legislation.

This case study relates to the field of collaborative editing and revision of complex legislative documents — specifically a method to allow multiple authors, working independently and in parallel to amend bills and automatically merge changes made by many authors to different copies of the bill back into a single version.

The application described in the study is known as *The Parliamentary Workbench* (PWB). PWB from Propylon Inc is an XML-based legislative solution suite. This study focuses on the Bill Drafting, Amendment and Engrossment modules within the PWB and outlines how the solution can be used to address many of the requirements of collaborative editing and revision.

Table of Contents

1. Drafting Legislation in XML

1.1. Summary

An important part (indeed the core) of the business of legislatures or parliaments is drafting, scrutinising, and changing legislation.

This case study relates to the field of collaborative editing and revision of complex legislative documents — specifically a method to allow multiple authors, working independently and in parallel to amend bills and automatically merge changes made by many authors to different copies of the bill back into a single version.

The application described in the study is known as *The Parliamentary Workbench* (PWB). PWB from Propylon Inc is an XML-based legislative solution suite. This study focuses on the Bill Drafting, Amendment and Engrossment modules within the PWB and outlines how the solution can be used to address many of the requirements of collaborative editing and revision.

1.2. Process Overview

The production of legislation production involves the submission and consideration of multiple sets of changes to documents. Typically the process begins with a base text to which multiple changes are then proposed. These changes are debated and then some changes are accepted, some rejected.

In order to assist the legislature's consideration of the various amendments it is necessary to describe the amendments and list them. Since there may be many hundreds of changes to documents and many of these changes may address the same area of the document it is difficult to show changes purely by marking the proposed changes on a single copy of the document. Instead formal descriptions of the changes are prepared known as *amendment lists* or *amendment instructions*. Typically, these give a list of self-contained descriptions of the proposed amendments. They are designed to be read alongside a clean copy of the bill as introduced. The instructions typically use descriptive wordings based on page and line numbers and on the semantics of the bill. For example:

- "In page 3, before section 4, to insert the following new section:"
- "In page 4, paragraph (c), line 7, to delete 'and' and substitute 'or'."
- "In page 14, line 29 to 25, to delete Subsection (2)."
- "In page 21, paragraph (b), line 25, to delete 'of' where it firstly occurs and substitute 'in'."

It is worth noting that the resulting descriptions are difficult to prepare and require much scrutiny to ensure their accuracy.

Note also that the process of scrutinising changes is not simply matter of considering each change in isolation and then agreeing yes or no. In the majority of legislatures one of the tasks of the scrutinizing committee is to organize the consideration process in such a way that mutually conflicting amendments (e.g., one amendment deleting a paragraph and another amendment modifying it) do not pass. Then amendments are grouped and considered together in such as way as to avoid this problem. Understanding how to group amendments requires an understanding of the possible interactions of the various amendments.

Once changes have been agreed the approved changes need to be merged into the text of the original bill in order to produce a modfied text which has been altered in accordance with the wishes of the legislature. The effect of the change on the text of the bill must match the change described in the corresponding amendment instruction. In addition, it may be necessary to show how text has changed by means of indicative typographic conventions such as underline, strikeout, bracketing or capitalization.

2. The Legislation Editor

2.1. General Characteristics

The Legislation Editor is based upon the OpenOffice.org Writer application and so, not suprisingly, looks and feels like a word processor. It includes all of the features that users have come to expect from a word processor including rich text display, spell-checking, multiple undo, and cross-referencing. It has been simplified to remove unnecessary features, minimizing the training requirement, while containing a range of custom features related to the drafting and amendment of bills.

In addition to its powerful array of word-processor functions OpenOffice.org Writer uses an underlying XML data-model which is clean, well-documented and easy to work with.

2.2. Color Coding

Color-coding is used to easily identify structural elements in the document.

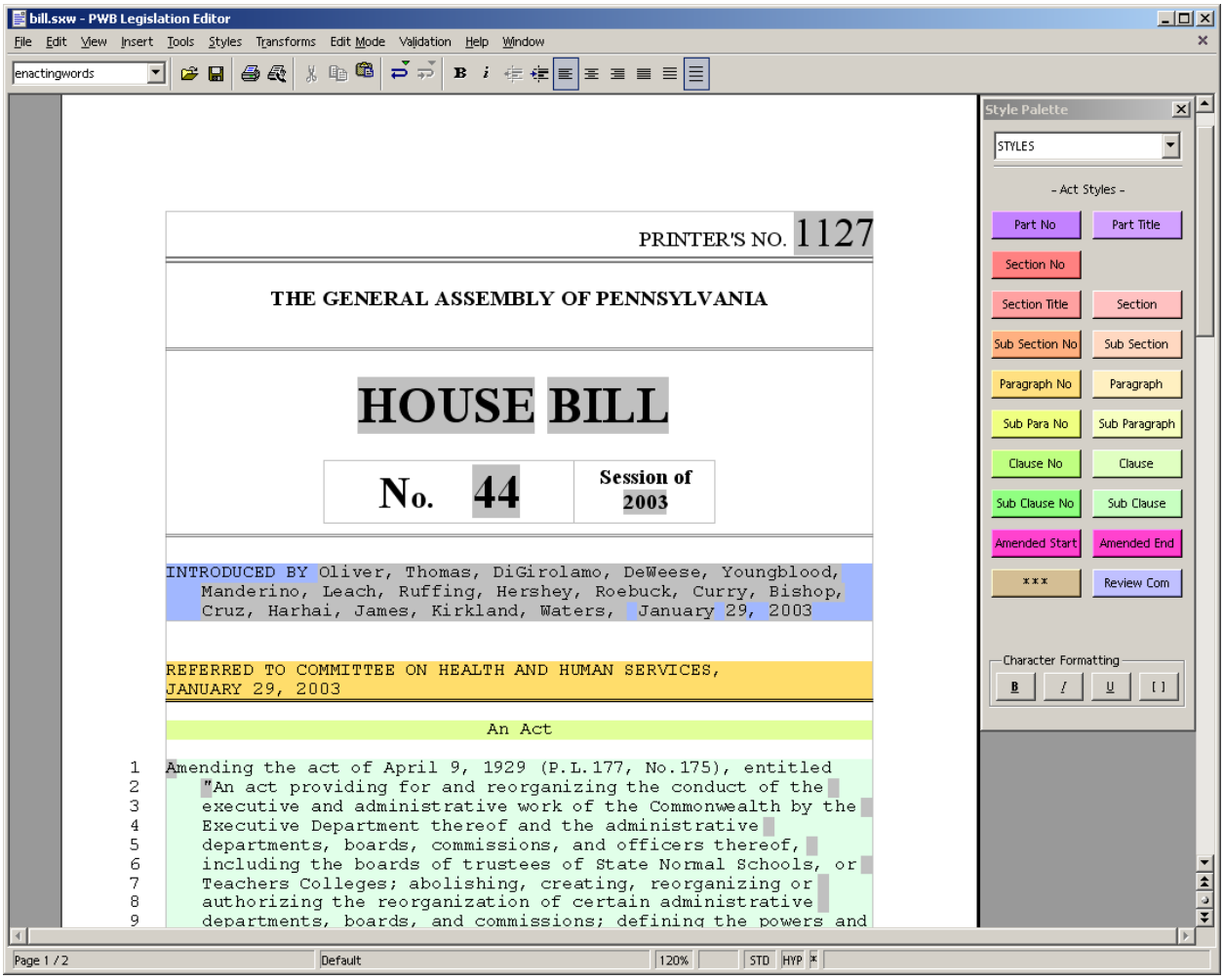

Figure 1. Color Coded Markup

2.3. Other Editing Modes

The system also provides other editing modes including a WYSIWYG presentation:

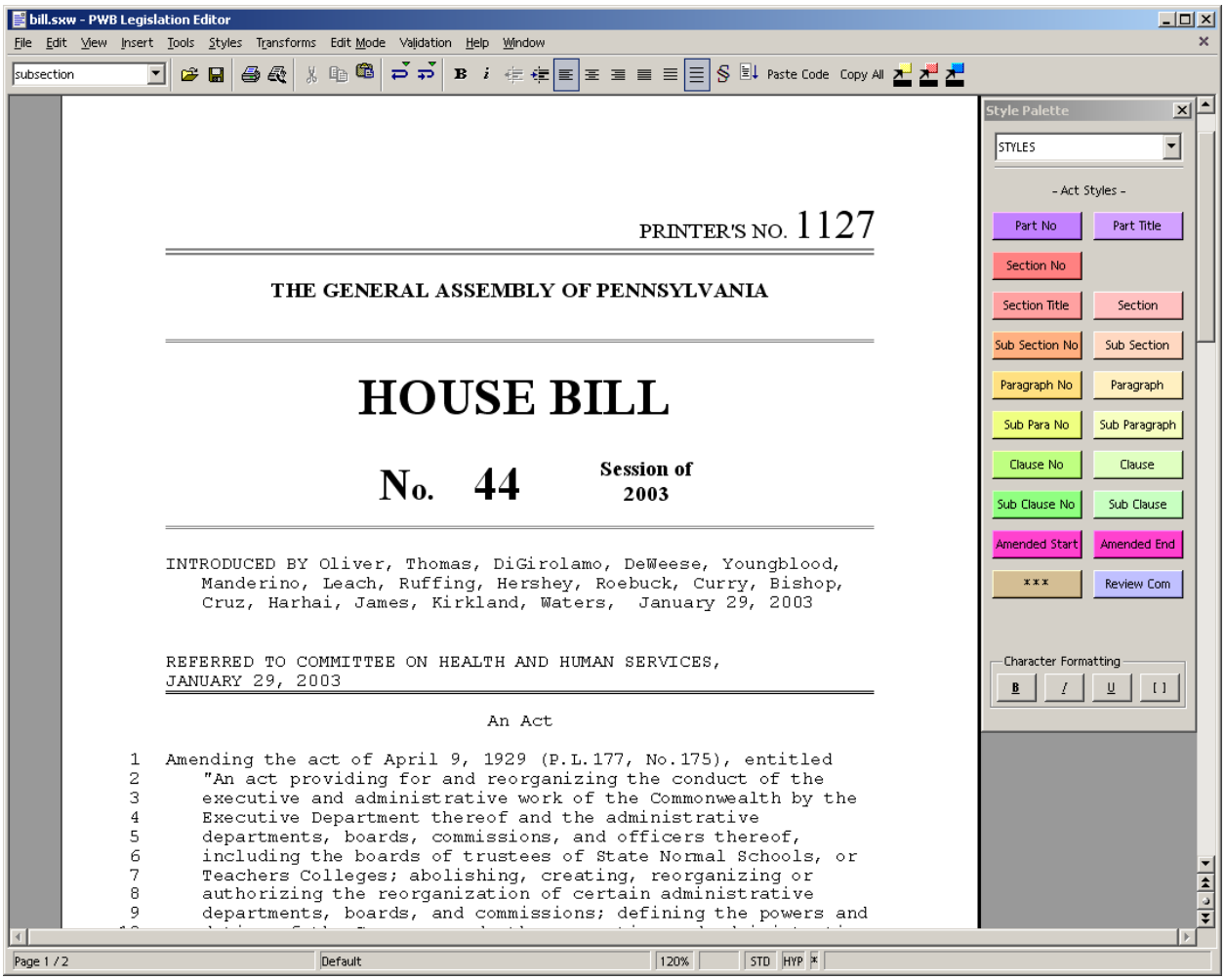

Figure 2. WYSIWYG View

2.4. Numbering

The system provides auto-numbering of even the most complex bills based on a legislature's existing numbering system. It has the ability to distinguish between;

- automatic numbering which will update automatically with the insertion and deletion of sections.
- fixed numbering which does not change during subsequent processing (this is often used when amending existing statute)

As well as section numbering, The Legislation Editor provides sophisticated line-numbering functionality to support the amendment process.

2.5. Manually-formatted Text

The Legislation Editor has the facility to embed manually-formatted (non-semantic) text within the document and reliably preserving that manual formatting though to print and web publication. This feature is a critical requirement for bill which need to include text that may not match the formatting of the local statute; for example, a fragment from a federal constitution or international treaty.

2.6. Validation

While the the Legislation Editor allows great freedom to drafters in they way they edit bills, it is critical that all documents stored in the legislative repository are properly structured in order to enable automated processing downstream. The Legislation Editor provides a validation feature to help a drafter identify and correct any potential problems. Validation performs an analysis of every element of the document and places accurate specific instructions for correcting problems in the document.

3. Amendments and Engrossing

PWB offers a number of features to automate the amendment and engrossment cycles of legislative documents. In this section the term “engrossment” is used to mean the incorporation of accepted amendments into a revised copy of the document at the end of each stage.

Individual legislatures vary in both the details of the processes, and in the names used to describe processes and stages, but most follow amendment cycles which are broadly similar.

At each stage in the process amendments can be submitted by legislators. Often, these proposed amendments are submitted in advance in writing, although many legislatures allow the submission of amendments “on the floor”; during session.

Amendments are typically offered as instructions relative to the last printed copy of the bill; for example “On page 23 line 14 to strike out the word ‘and’ and replace it with ‘or’”.

Amendments that are offered during each stage are usually printed and circulated to the relevant legislators to assist in deliberations. Because proposed amendments are submitted from a wide variety of legislators, it is possible that many will contradict each other.

Near the end of each stage, a subset of the proposed amendments is accepted. This selection process can be a committee decision or a vote. A revised copy of the bill is printed and circulated that contains the accepted amendments, and that is forwarded to the next stage of deliberation.

Most legislatures have a statutory requirement to print and circulate revised copy of the bill rapidly — often the following day. This can place a tremendous burden on legislative staff, because an extensive process of manual cutting and pasting (or data re-entry) and manual formatting is required to apply the accepted amendments. If external typesetters or printers are used, additional onerous proof-reading cycles are involved.

PWB automates the amendment and engrossment cycle by enabling all proposed amendments to be recorded quickly and easily, and automatically incorporating accepted amendments into the next-stage version of a bill.

3.1. Recording Amendments

Other legislative automation systems require users to use a complex and time-consuming series of forms and data-entry screens to enter amendments. The drafter may need a series for drop-down menus (or a wizard) to identify the Section, Sub-section, Paragraph that they propose to amend. They often enter significant additional metadata to identify the type of amendment they are about to enter. They must then type in the new revision, often using special codes for

formatting. Finally, the drafter must do a “print preview” to ensure that the proposed revision matches the intent of the legislator.

PWB makes it very simple to enter proposed amendments into the system. The drafter selects a bill to amend, selects the submitting legislator, and clicks “Add Amendment”. Depending on the requirements of the legislature, the drafter can be prompted to enter additional meta-data (for example, the date and time the amendment was submitted).

PWB then places the drafter directly into a copy of the bill where they make the proposed amendment directly to the bill using a traditional word processing “track-changes” view. Deleted text is shown as stricken out and inserted text is shown underlined. Other forms of change markup can be used if desired.

When the amendment is completed, the drafter selects “Complete Amendment” and PWB analyses the changes made to the document. From the changes, PWB generates an instructional amendment and presents it to the drafter in a dialog box for approval.

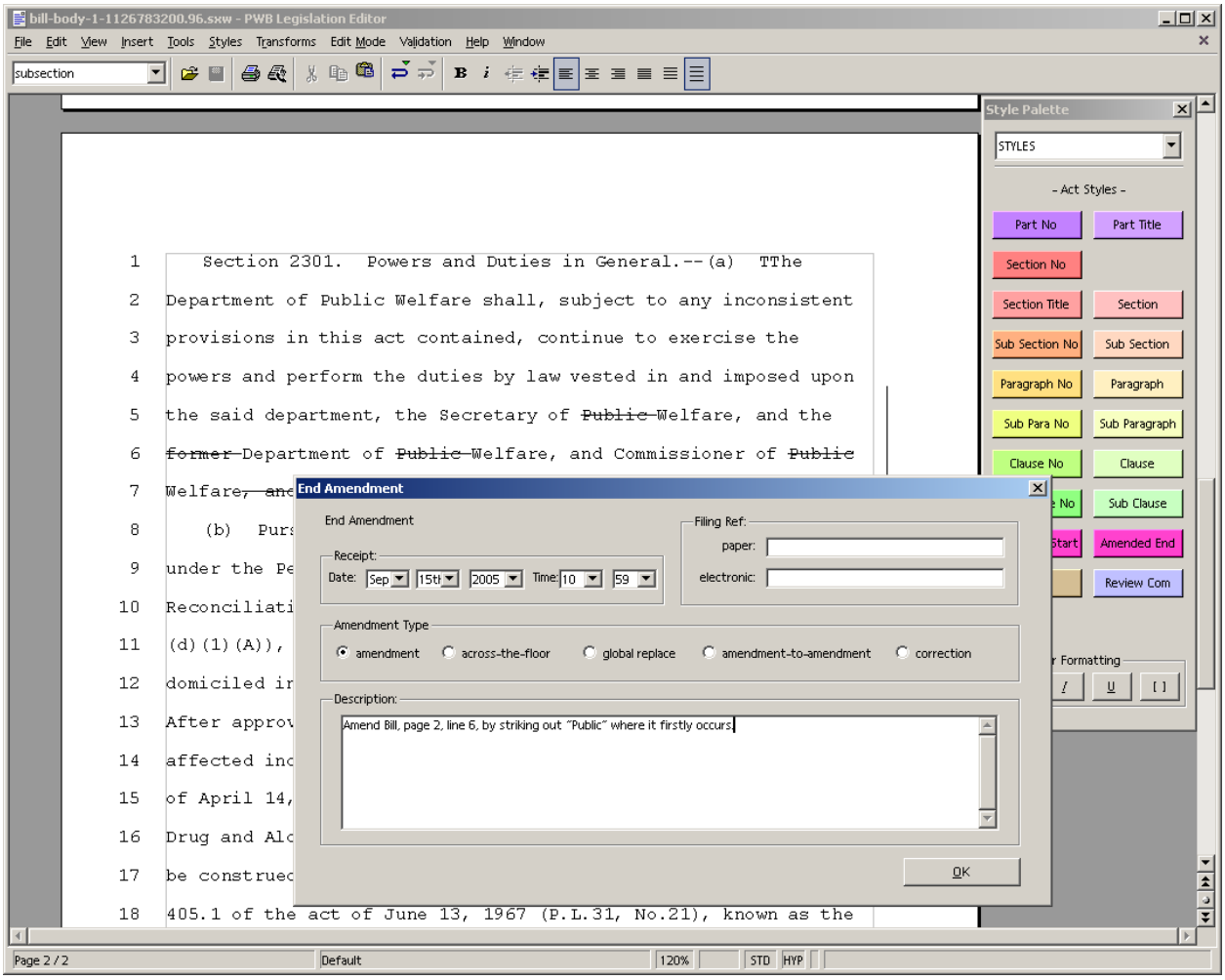

Figure 3. Generating an Amendment Instruction

PWB contains a configurable set of rules that allow it to accurately describe the amendments in the proper legal language used by the legislature.

```
     S0014B1283A157     Clyde Hatter     15/09/05          #90                A157


                       AMENDMENTS TO SENATE BILL NO. 14

                                          Sponsor: REPRESENTATIVE, ARGALL

                                                Printer's No. 1283


1       Amend Bill, page 2, line 5, by striking out "Public".

2       Amend Bill, page 2, line 6, by striking out "former".

3       Amend Bill, page 2, line 6, by striking out "Public" where it
4   firstly occurs.

5       Amend Bill, page 2, line 6, by striking out "Public" where it
6   secondly occurs.

7       Amend Bill, page 2, line 7, by striking out ", and the former
8   Department of Welfare".
```

Figure 4. Amendment Instructions

The automatically generated instructional amendments are made relative to the last printing of the bill. The system has the capability to simultaneously store multiple contradictory potential amendments.

By capturing amendments in the context of the bill, PWB delivers a range of unique benefits. Not only is it much simpler to learn and use, it is so fast and unambiguous that that it can be used in-chamber to capture oral amendments offered across the floor.

It also offers new possibilities for legislators. Other legislative automation suites only allow legislators to see amendments in context after they have been accepted and engrossed into a bill at the end of each stage. PWB allows legislators to view proposed amendments in context during deliberations. It also allows the automatic generation of a variety of reports (by print or web) to assist legislators during deliberations.

3.2. Engrossing

At the end of each stage, those amendments that have been accepted must be engrossed into the next stage version of the bill. PWB supports this process by capturing the votes on each amendment.

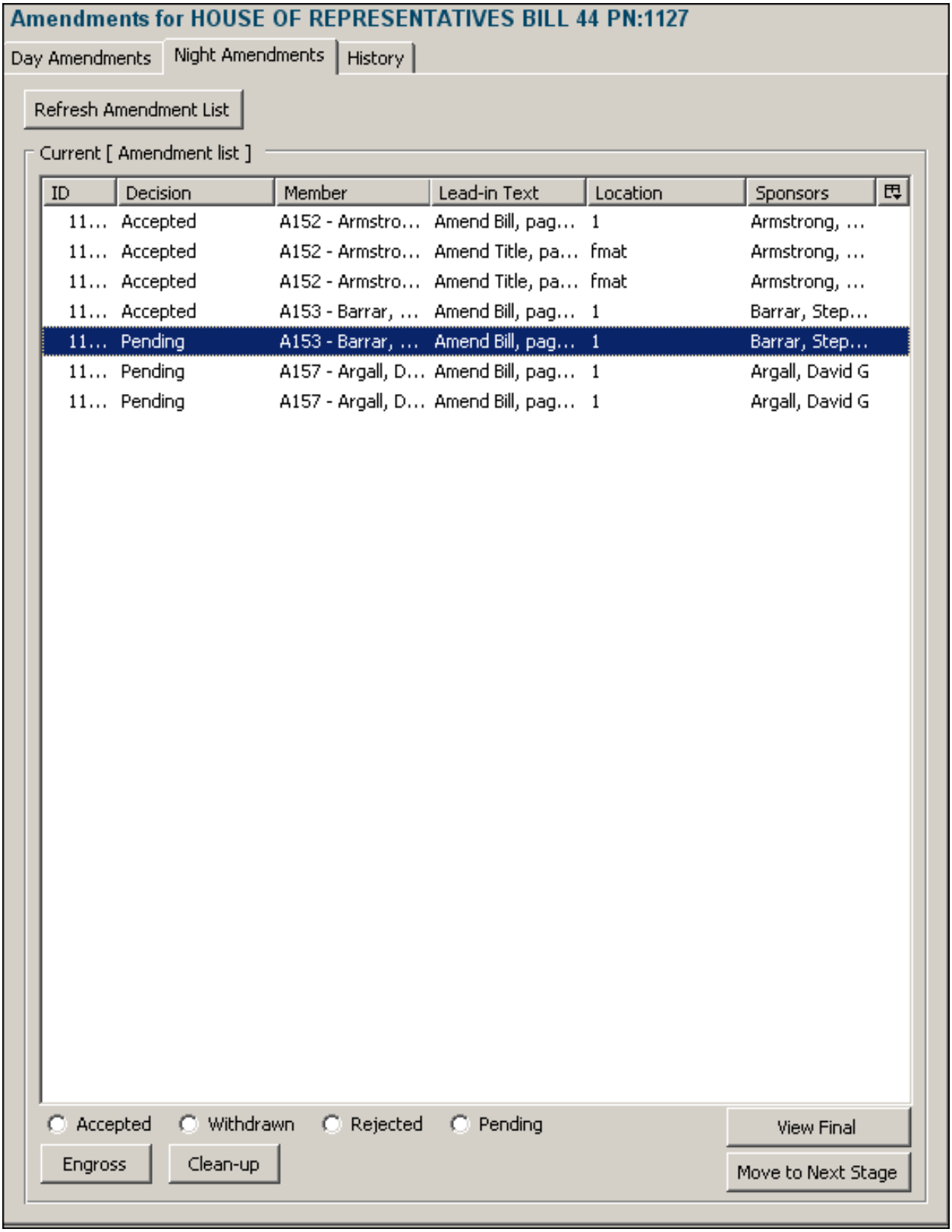

Figure 5. Capturing Amendment Decisions

PWB provides a screen that displays all the amendments that have been proposed during that stage. Based on the decision, each amendment can be accepted or rejected. By pressing a single “Engross” button, the system generates a new version of the bill that contains only the proposed amendments.

In almost all cases the accepted amendments will be logically coherent. In the event that two or more of the accepted amendments are mutually exclusive, (an attempt is made to change the same text in two conflicting ways), PWB presents a dialog to the drafter to enable the conflict to be resolved.

A new version of the bill is presented to the drafter with the accepted amendments highlighted in the bill. By rolling-over each changes, the drafter can see meta-data about the change.

3.3. End-Of-Stage Processing

The protocol for removing change markup varies across legislatures. Some legislatures preserve all change markup right through bill signing, and only remove it when the bill is consolidated into code. Other legislatures remove the change markup at the end of each stage (presenting a new “clean” version of the text to the next stage). There are a wide range of options in between.

PWB can be configured to support any required process. It stores the source of each amendment (including the chamber, party and name of the introducer) and can use that information to change how different amendments are displayed and printed. Apart from color-coding, the system can display amendments in many ways including underline, stikeout, double underline, double strikethough, italic or bold, or in different fonts depending on the origin of the amendment. It can also handle “amendments to amendments” in a variety of ways.

For legislatures who start each stage with a “clean” bill — i.e. with the underlining removed and stricken text completely eliminated from the bill, the PWB offers a simple process to remove the markup. Selecting “Accept Changes” presents a dialog box that lists all of the changes that have been accepted at that stage. The drafter simply accepts them individually or collectively.

In addition to variations over the handling of change markup, legislatures differ in when they renumber sections in a bill. PWB can support any desired process for section renumbering.

The final step at the end of a stage is to render the next-stage bill for printing and circulation. It is at this point that new line and page numbers are typically generated. During the rendering process, PWB extracts the line and page numbers from the rending engine, and re-incorporates those into the canonical XML file. This enables the cycle of amendment tracking to start again based on the newly printed document.

This entire Amend–Engross–Publish cycle is typically repeated several times in each chamber. Depending on the statutory powers of the executive branch within the jurisdiction, there may be additional stages of amendment processing prior to signing.

4. Print

Print output is particularly important to the functioning of a legislature because amendments are offered at each stage relative to the last printed copy of the bill. Ensuring accurate line numbering of print output is of paramount importance.

PWB can render complex line numbering in print output. As part of the rending process, it also captures and stores information about the line numbering in the editable document. Incorporating this information into the canonical XML files enables the automatic generation of instructional amendments.

In legislatures where line and page numbering are generated by an external entity (for example a Government Printer using a specialized typesetting technology), PWB can be configured to import the externally rendered files and extract line and page numbering.

5. Notes on the Legislative Editor

The Legislative Editor enables drafters to quickly and easily creation of structurally correct, properly formatted legislative documents that can be automatically processed in many different ways.

One of the most obvious approaches to creating complex structured documents is to use an XML editing tool. While this approach guarantees well-structured documents, it comes at the expense of a high training requirement and a great deal of frustration to drafters.

Because XML editors enforce strict structure at all times, the drafter must consistently move from one valid state to another. For example, dragging and dropping a set of text from one part of a bill to another in an XML editor is likely result (temporarily) in an invalid document, and most XML editors would prevent the user from doing so.

XML editors typically require users to learn tens or even hundreds of “macros” to accomplish basic tasks. Not only does it take a drafter months to become proficient, but the drafter’s attention is always divided between the constraints of the editor and the actual language of the legislation.

The Legislative Editor eliminates this burden, and is immediately accessible to anyone who has used a word processor. Notes on the criteria used by Propylon to evaluate and select a drafting tool can be found in section .

The goal of the Legislative Editor is to shield the drafter as much as possible from the constraints of the underlying data model. It achieves this by presenting a true word processor interface that allows drafters to create, edit, and amend legislative documents using a natural authoring style.

5.1. Design Goals

The design goals of the The Editor module are to

- Enable the drafter to quickly and easily create legislative documents that are structurally correct, enabling reliable automated processing later.
- Be highly customizable so that the editing experience can be optimized for each type of legislative document.
- Eliminate re-keying of data by enabling automatic insertion of text fragments or document elements stored anywhere in the Repository.
- Enable the drafter to draft and amend bills with little or no consideration for the complex formatting of the final print output.
- Provide automated section numbering, and automatically update cross references when required
- Reduce the training requirement to around a day, and enable a drafter to become expert in a week.
- Accommodate the needs of drafters with different interface preferences, including font size, color-coded vs. black & white editing, and keyboard-only vs. keyboard & mouse editing
- Enable a drafter to work “offline” from the Repository. This allows legislative staff to work in committee rooms or at home, and enables external agencies to create and submit draft bills if the legislature deems it appropriate.
- Be easily deployable across a wide range of platforms and IT environments, and provide centralized administration of desktops.

The Legislative Editor is suitable for a wide range of authors because it offers several editing modes. Authors can choose between a WYSIWYG view and the color-coded view which differentiates structural levels within the document.

Drafters can select functions and styles using a mouse or can work with minimal dependence on a mouse using function keys or keyboard shortcuts.

For highly structured documents like legislative bills, color-coding of dense text has two huge benefits. Firstly, it makes it easy to glance through large volumes of text, picking out the vital structural information without requiring a close reading. For high volume, fast turnaround publishing environments, this is a real boon.

Secondly, it provides a simple, non-intrusive way to enable drafters to tag the content they create based on what it means — semantics. Compared to the traditional tag-based route to document semantics the majority non-technical people prefer the simplicity and clarity of color-coding.

5.2. Validation

While the Legislative Editor allows great freedom to drafters in they way they edit documents, it is critical that all documents stored in the legislative repository are properly structured in order to enable automated processing down-stream. The Legislative Editor ensures that the bill has correctly formed though a Validation function.

Validation is a menu item in the Legislative Editor that can be clicked by the drafter at any time during editing. It is run automatically when a document is checked back into the Repository. Validation performs a sophisticated analysis of every element of the document and detects any errors that could prevent automated processing of the document later on. Errors identified by Validation are placed in the document with a black background directly above the error and contain a description of the problem and a suggestion for fixing it.

Validation analyzes both the structure and the content of any element of the bill, and can contain sophisticated rules to ensure drafting consistency across all bills. The following sample Validation Rules, taken from the The Editor, give an example of the power and flexibility of this feature.

- A bill must start with BILL-REF, followed by an INTRODUCERS, SHORT-TITLE, LONG-TITLE, and ENACT-CLAUSE
- BILL-REF must consist of the words “House Bill No.” or “Senate Bill No.” followed by a 4-digit number.
- Every name in the INTRODUCERS section must appear on the SERVING-LEGISLATORS list (stored in Repository)
- Every SHORT_TITLE must end with a 4-digit year.
- ENACT-CLAUSE must be followed by a PART or a SECTION
- Only a PART, a SECTION, a SUB-SECTION or a PARAGRAPH are allowed follow a SECTION
- Every PARAGRAPH must end with one of the following; “.” “,” “and” “or”.

Biography

Lisa **Richards**
Senior Engagement Manager
Propylon Inc. [http://www.propylon.com]
New York
New York
United States of America

With more than 15 years in the industry, Richards provides a strong portfolio of skills in marketing, strategic business planning and development, operations, customer service management, and project management.

Richards founded CD/LAW, a legal publishing business that was eventually acquired by Lawyer’s Cooperative Publishing, a subsidiary of Thomson International, now West Group. She's been with several of the leading XML vendors over the years, and is now employed by Propylon and leading the Bill Drafting Project for the Pennsylvania General Assembly.