



CALTECH/MIT VOTING TECHNOLOGY PROJECT

A multi-disciplinary, collaborative project of
the California Institute of Technology – Pasadena, California 91125 and
the Massachusetts Institute of Technology – Cambridge, Massachusetts
02139

A BALLOT DESIGNER AND BALLOT DESIGN CRITIC Undergraduate thesis

Elizabeth Herbert
MIT

Key words: ballot design, 2000 presidential election, butterfly ballot

VTP WORKING PAPER #52
September 2002

Introduction

The weeks following the 2000 Presidential Election were filled with uncertainty as Americans waited to hear the outcome of the Florida recounts and find out who would be their next president. People across the country were reminded of the importance of even a single vote. Unfortunately, however, many of the votes cast in the election were not counted or were counted for the wrong candidate. In fact, an estimated 4 to 6 million votes were lost due to problems such as confusing ballots, mix ups with registration, and faulty equipment¹. Unclear ballot designs caused many voters to submit ballots with overvotes, undervotes, and unintentional votes.

The most notorious example of confusing ballot design is the infamous Palm Beach County butterfly ballot. Post-election analysis showed that many of the votes that Buchanan received were mistakes. The Palm Beach Post surveyed more than half of the registered voters in one precinct and found that none of the voters surveyed intended to vote for Buchanan. Furthermore, some of the respondents indicated that they realized they had mistakenly voted for Buchanan as they were leaving the polling place². This suggests that the ballot was confusing and that many of the other Buchanan votes may have been mistakes that went undetected. The main problem with the butterfly ballot was that although Gore was listed as the second candidate on the left side of the ballot, the corresponding circle to punch fell third in the column. The second circle was supposed to be for the top candidate on the right side, Pat Buchanan. In a county with many elderly voters who have poorer vision and are slower to process information than their younger peers, it is understandable that this sort of confusion could occur. The net effect of this confusing design was a ballot biased against Gore and towards Buchanan³.

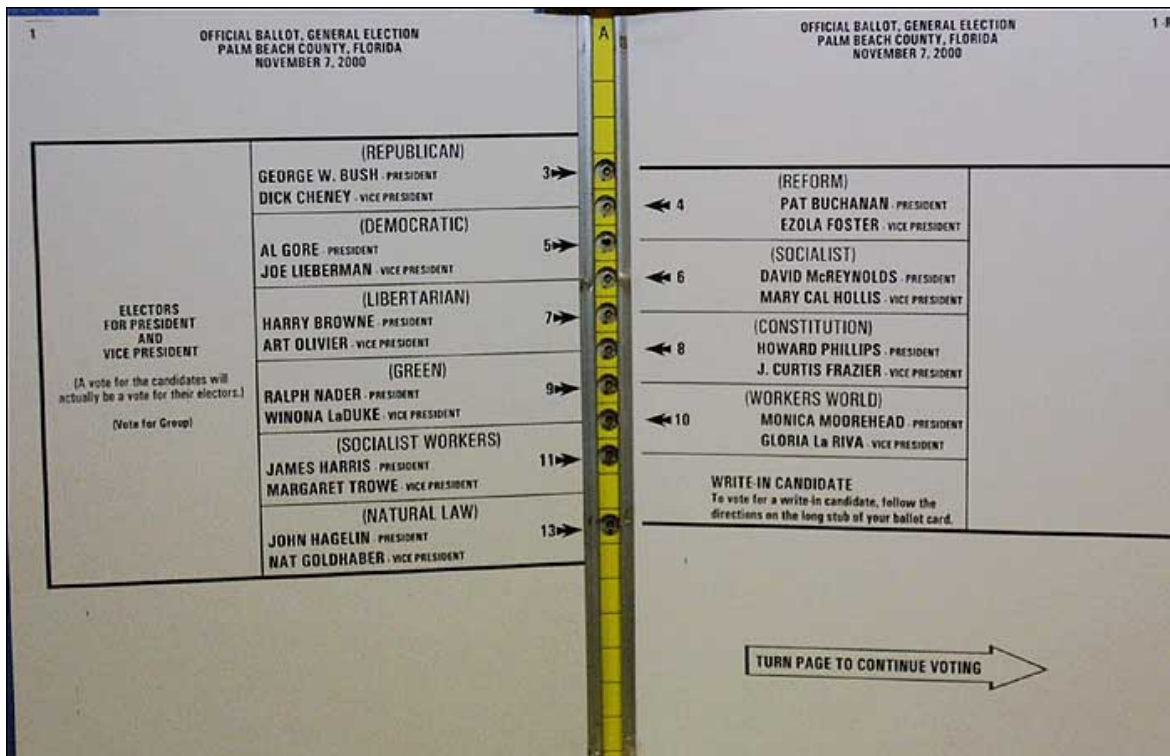


Figure 1: The butterfly ballot used in Palm Beach County caused confusion when some voters intending to vote for Gore punched the second circle in the column instead of the third. This resulted in an atypical large number of Buchanan votes in this county as compared to similar counties.

In other areas of the country, voters submitted incomplete ballots because they failed to notice that the ballot was two-sided. Some voters correctly completed their ballots but ended up feeding them backwards into a punchcard machine, rendering their votes illegible. In addition, ballots created on what is known as a full-face electronic DRE machine are a likely cause of voter confusion when the number of candidates and issues becomes large since this forces the print size to be extremely small to conform to a specific area on the machine⁴.

In some areas of the country, language ambiguities are to blame for voter error. The most common source of this confusion is the phrase “Write-in” because many people write-in their choice after they have already checked the appropriate box. Since these ballots fail to clarify that the write-in slot is only for candidates who do not appear on the list, people end up voting for the same candidate twice, generating an overvote that does not count at all⁵. Language used for ballots must be carefully chosen so that this sort of confusion does not exist.

Voter error due to ballot design must be eliminated or at least significantly reduced before our voting system can be considered fair. To accomplish this, new standards of ballot design must be established and put into use.

Current Systems of Ballot Design

Los Angeles County

L.A. County officials use a software system that was developed in-house to design their ballots. This system only designs punchcard ballots as these are the standard in all of the precincts for which these officials are responsible. Also, the system only prints ballots in one language, namely English. If voters do not speak English, they are permitted to bring a sample ballot form into the voting place that they can use to translate the English ballot into their own language. This way of handling multiple languages simplifies the software requirements. The software system works by reading in a file of parameters and calculating an appropriate layout. Design flexibility is severely limited because most of the layout decisions are specified by federal, state, or local election codes. However, ballot designers do have some flexibility in decisions such as spacing

between contests or between candidate names. To make changes to the layout, the ballot designer must figure out what the new parameters should be, change them in the parameter file, and rerun the software with the new file as input. All changes must pass a thorough review process before being approved.

The most serious drawback of the software program used in LA County is that ballot designers cannot see the layout as they are making changes. Since the program lacks the convenience of a drag-and-drop user interface, the only way that the layout can be updated is by rerunning the software with a new parameter file. The ballot maker outlined in this paper offers this same type of input option, but it improves upon the LA County system by offering alternate ways to change the ballot: a drag-and-drop graphical user interface and a toolbar containing drawing tools.

Danaher Controls

The Guardian Election Management Software (GEMS) marketed by Danaher Controls is similar to the ballot maker in several ways. In Danaher's system, the first step in designing a ballot is to enter all the names of candidates and offices. This is analogous to creating the XML file that specifies the contents of the ballot in the software discussed in this paper. After the programs have the necessary candidate and race information, both programs create a drag-and-drop graphical user interface that the ballot designer uses to layout a ballot. In the Guardian system, however, a lot of the drag-and-drop is done by codes to save space on the screen and allow the designer to see a bigger part of the ballot at a time. This space-saving technique is important since the Guardian system can support as many as 504 fields for candidates, races, or initiatives. This feature could be

added to the ballot maker if large numbers of candidates and races become difficult for designers to handle.

The main drawback of GEMS is that it only works with one system, specifically a voting system marketed by Danaher Controls. In contrast, the ballot maker will be able to work with a variety of systems because it can either output an xml file that specifies the layout of a ballot or print a physical copy of the ballot for paper-and-pen or punchcard voting. This means that that a software system such as the ballot maker will be useful to a larger number of precincts without requiring that they update all their equipment at once. Another advantage is that it gives precincts the flexibility of choosing from different voting machines or even creating their own voting software since the XML file can be used as input to any program that knows its format.

Goals of Ballot Design

The ballot maker aims to make it easy for users to design ballots that are clear and that give fair treatment to all candidates and races. The software will use embedded knowledge about principles of good ballot design to guide users through the process of creating a layout whose features are arranged in a logical way and can be easily distinguished. Design decisions such as font type, font size, color, component size, alignment, order of components, and arrangement of components are crucial choices that determine the appearance and usability of a layout.

A design that is fair must eliminate bias that can result from position effects, size effects, and contrast effects. For example, in many precincts, the incumbent gets the privilege of having the top spot on the list of candidates, a clear advantage on election

day. Precincts attempting to eliminate position effects use methods such as randomization of candidate names across different ballots or random drawing for an order that will be constant across all ballots. Another example of layout bias is that races of greater importance are generally placed ahead of races of lesser importance on the ballot. Especially in cases where the ballot is multiple pages long, voters are more likely to leave an item blank if it is towards the end of the ballot. This problem actually became quite serious in precincts where two-sided ballots were used because the second half of the ballot was left blank by many voters who failed to notice the races on the back. While it will always be possible for voters to skip items, the layout should attempt to make voting on all the items as quick and easy as possible so that people will complete most or all of them before deciding to turn in their ballot for counting. Similarly, all items should be of sufficient size and contrast so that they are not skipped over by mistake.

Regardless of the layout of a ballot, voter accuracy is greatest when voters can easily see what items they have completed and what items are currently overvotes or undervotes that will be excluded from the count. This sort of local feedback is a crucial feature of a well-designed voting system⁵. Feedback should be immediate so that the most current state of the ballot is always visible to the voter. The visual display should be clear so that a voter can easily see what items are complete and what items are still undervotes or overvotes. Also, he should be able to clearly see the choices that he has made on completed items.

A voter should have the option of removing or changing a vote if he has made a mistake or simply changed his mind. The method for updating the ballot should be obvious, providing logical and clear ways to remove or change a vote. By allowing

voters the opportunity to go through this review process, mistakes can be kept to a minimum. Furthermore, voters can leave the polling place feeling confident that they correctly completed the ballot.

Architecture

The ballot maker resembles a traditional paint program where the user creates or modifies a design using a toolkit and a drag-and-drop interface. The software creates an initial layout through three basic methods: by using rules specified in an XML file, by using specific formatting instructions contained in an XML file, and by using knowledge about "best of breed" user interfaces. Then, the user modifies this layout using the features available on the toolbar or by dragging-and-dropping components to change size or arrangement.

The ballot maker supports rules that specify how to arrange the components in the ballot including properties such as size, location, font, and appearance. These rules can specify uniformity across all the components or can specify, for example, that candidate boxes increase in size or in contrast as you look down a list. The XML string containing the rules can be read in from a database or directly included in a Java file. This makes it simple to make changes to the layout rules.

In addition to supporting rules, the ballot designer can read an XML file that specifies precise values for various formatting parameters. These parameters include component features such as location on the screen, size, color, font, and appearance. Like the rules file, this file can also be read from a database or coded into a Java file to facilitate additions and alterations to the formatting specifications.

In the event that the ballot maker is run without a fully-descriptive rule file or format file, the software will call upon its knowledge of good design principles to make default layout choices. For example, studies have shown that position effects can lead to bias when making a selection. In fact, election laws often take this effect into account when they specify that the incumbent candidate get the top position in a list. Possible ways of counteracting this effect include making the choices increase in terms of button size, font size, font darkness, or button contrast. This knowledge about principles of user interface design will not override specifications made in format or rule files, but it will be used as a default case if this data is omitted from the files.

Once the layout decisions from files and default settings are made, the user can manipulate the ballot through a drag-and-drop interface. Through this interface, the user can resize, re-order, or move various components. In addition to drag-and-drop, the user can alter the appearance of the ballot through buttons on a toolbar. These buttons cover functions such as add line, add graphic, add text, or add new contest or issue.

Once the user is satisfied with the layout, he has several options. He can save the appearance into format and rule files for later revision or for input into an electronic voting machine. Another alternative is to print the design into paper ballots or punchcard ballots. All of these methods can be invoked using the toolbar.

Modules Used by the System

Ballots are divided into two parts: the main ballot area and a table of contents. The main area contains races and initiatives as well as a header that specifies precinct information and the date of the election. The table of contents is an area that will keep

voters updated on their progress while working on a ballot. This table is a way for voters to quickly and clearly see which races they have completed and what choices they have made. It also indicates overvotes and undervotes to make it easier for voters to detect and correct mistakes.

The main ballot separates its components into three distinct categories: `ballot`, `ballot_item`, and `ballot_item_option`. The `ballot` category represents the entire ballot and requires input values for the election name, county, precinct, and date. It also requires a unique election ID number. The `ballot` has an option for specifying the language being used. Ballots are made up of any number of `ballot_items`.

Each `ballot_item` represents a specific race or initiative⁷ within the election. It requires input values for the item name, an ID number, whether the item is an issue as opposed to a race, whether the item receives preference in placement order on the ballot, and how many selections are allowed. Each `ballot_item` can contain `ballot_item_options`.

The `ballot_item_options` are the actual choices that can be selected in a race or on an initiative. Input values are required for the option name and option ID. Also, an option can hold a value to show that it selected. If this value is not specified, the option defaults to being unselected.

The table of contents frame is a separate part of the ballot that provides local feedback to voters. This feedback informs voters as to which ballot items are complete, which ones are undervotes, and which ones are overvotes. In addition, for the completed items, voters are able to see their choices which makes it easy to detect mistakes. The ballot maker allows the designer to layout how an item should look in the table of contents and how to distinguish a completed vote from an undervote or an overvote.

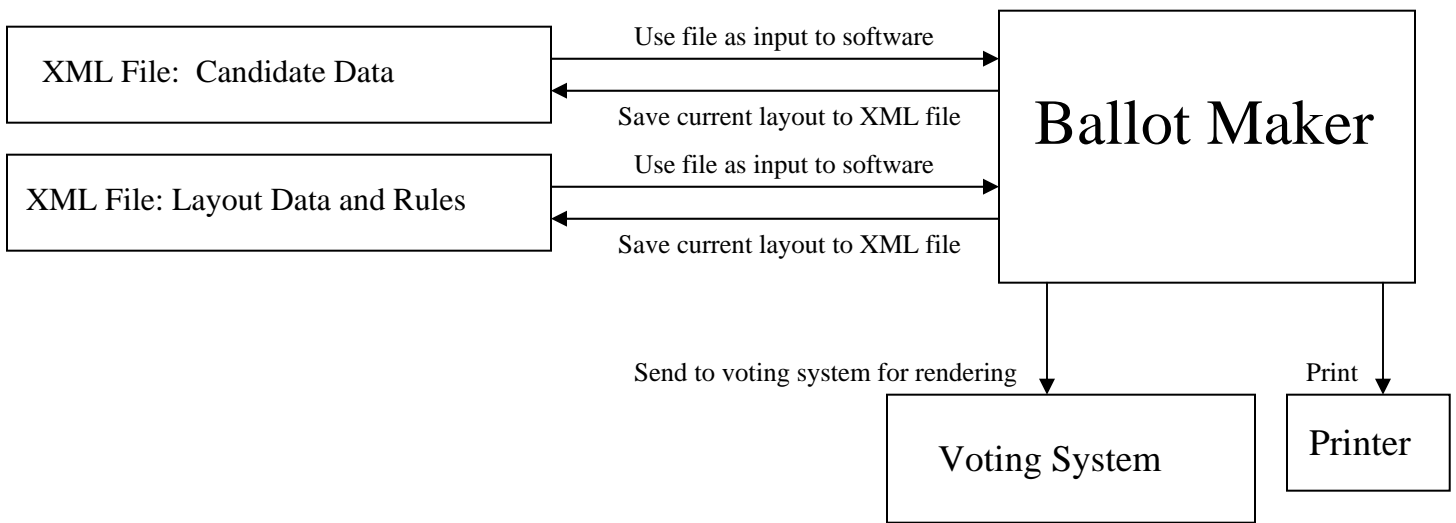


Figure 2. Flow chart showing where the ballot maker gets input and what output options it offers.

How the Software Works

At start-up, the ballot maker, coded in Java, constructs three frames: the toolbar frame, the ballot frame, and the frame for the table of contents. The program checks for XML files describing the content of the ballot or any formatting instructions for the layout. If either file exists, the program runs through its tags, extracting the necessary information.

For each ballot item found in the ballot description, the software draws a section on the ballot according to the specifications given in the format file. If this file does not exist or is incomplete, the software calls upon an embedded set of layout specifications to fill in the gaps.

In addition to drawing a ballot layout to the screen, the software draws the table of contents and a toolbar consisting of drawing tools, a save button, and a print button. The

drawing tools are the same each time you run the program, but the appearance of the table of contents depends on the information contained in the XML files.

After the three frames are drawn to the screen, the ballot designer can drag-and-drop the various components, and the software will respond by repainting the appropriate area. At any time, the user can press the save button to write the current states of the ballot and the table of contents to XML files. Also, he can press the print button to print out a copy of the design.

Interaction with Other Systems

Any ballot design can be saved in the form of two XML files: one containing the candidate data and one containing formatting instructions and rules that govern the layout. These files can then be used as input to any software or system. Currently, the only system that has been created to use these files as input is the voting system created at the MIT Media Lab by Ted Selker and Jonathan Goler. This system renders the XML text onto the screen and then handles all remaining aspects of voting, including counting the votes and ensuring security.

Discussion

In programming the software for the ballot maker, there is a trade-off between giving users total control and enforcing good ballot design principles. In some areas such as resizing the various candidate boxes in a race, the user only has restricted control and cannot make the boxes random different sizes. For instance, if a user tries to make the candidate at the top of the list have a bigger box than the rest of the candidates, the

software will automatically resize all of the boxes to eliminate visual bias. However, other decisions such as the order of candidate names in a list are left completely up to the user. This creates somewhat of a danger though in that ballot designers will need to be trustworthy. If this software were used for official elections, it would be a good idea to have a review process for all ballot layouts before they are approved. This would reduce the power of the ballot designer to make choices that might influence the outcome of an election.

Future Work

The most important feature of the ballot maker is that it contains knowledge about “best of breed” ballot layouts. Therefore, the software can be enhanced by improving the quality and extent of this content. This knowledge should be drawn from studies that test how various aspects of the layout affect the clarity and simplicity of the design. If gaps exist in the research, new studies should be conducted to fill them.

Another important piece of the system is the user interface used to layout a ballot. This must be easy to understand so that ballot designers will not have trouble using the software. The goal of the program is to facilitate the ballot design piece of the election process, not to make it more difficult. Therefore, it is important to perform usability testing on the software to see what features should be changed or added.

References

¹Caltech/MIT Voting Technology Project. *Voting: What Is, What Could Be*, July, 2001.

²McLachlin, M. *How many Buchanan supporters are there?*, The Palm Beach Post, November 12, 2000.

³Brady, H. *Report on Voting and Ballot Form in Palm Beach County*, November, 2000.

⁴United States General Accounting Office. *Elections: Perspectives on Activities and Challenges Across the Nation*, October, 2001.

⁵Selker, T. "User Interface and Ballot Design as Part of an Improved Voting System," May, 2001.

Fischer, E. *Voting Technologies in the United States: Overview and Issues for Congress*, March, 2001.

Foote, E., and Smith, J. *Revitalizing Democracy in Florida: The Governor's Select Task Force on Election Procedures, Standards and Technology*, March, 2001.