A VVSG-derived model of election data

David Flater, National Institute of Standards and Technology

2009-09-08

Abstract

An effort to define a common data format for voting systems should begin with a data model that specifies the relevant concepts. To accelerate adoption and minimize conflicts, such a model should define the smallest set of concepts needed by the desired functionality. We present a small data model that suffices to cover the election definition and vote data reporting requirements of VVSG 2.0, with the exception of reporting for ranked order contests. This model may be used as the basis for continued work, in whole or in part, without restriction.

1 Introduction

The NIST Common Data Format Workshop was organized to identify and agree upon a set of requirements for a common data format for voting systems. For the most part, those requirements will follow from the goals that are identified for the format. However, regardless of what specific goals are chosen, we anticipate agreement that a common data format should be based on a coherent data model with strong conceptual integrity. In addition, to the extent that support for the common data format could potentially someday become a requirement for voting system certification, the format should support all of the voting variations defined in the Voluntary Voting System Guidelines (VVSG) [1].

In support of those objectives, we present a small data model that suffices to cover the election definition and vote data reporting requirements of VVSG 2.0, with the exception of reporting for ranked order contests. While the VVSG's lone requirement on ranked order reporting (Part 1 Req. 7.8.3.3-D) improves on the complete absence of such requirements in previous versions of the VVSG, it still leaves much unspecified. A supporting data model would need to evolve in conjunction with requirements as ranked order voting becomes more prevalent and applicable practices evolve.

We present the data model in two parts. The first part, in Section 3, is a minor revision of the data model that was used in Votetest [2] to structure test data for use in VVSG conformity assessment. It covers election definition and votes but not reporting. The second part, in Section 4, is an extension of the Votetest model to cover vote data reporting for non-ranked-order contests. Section 5 follows with description of some interesting design decisions that impacted the model.

2 Assumptions

This paper adopts and extends the terminology of VVSG 2.0 [1, Appendix A].

All entities in the data model are implicitly scoped by an election. It is assumed that different elections are stored in different databases, and any reuse of definitions from one election to another is accomplished by copying over the relevant data.

The data model is constructed from an integrated, top-level viewpoint. In practice, different portions of a voting system and different steps in the voting process will deal with only a portion of the data at any given time. It is expected that users of the data model will project and extract data from the integrated model as needed to support these limited viewpoints. Conceptual integrity is supported by traceability to the integrated model.

3 Election definition and votes

The data model is described in Figure 1 by a Unified Modeling Language (UML) class diagram [3]. Following sections explain the diagram.

3.1 Basic data types

BallotCategory (CodeList, a.k.a. "extensible enum") Arbitrary tag that may be applied to Ballots; e.g., Early, Regular, InPerson, Absentee, Provisional, Challenged, NotRegistered, WrongPrecinct, IneligibleVoter, Blank. Categories are jurisdiction-defined but are likely to include several classes of provisional.

Boolean True/false data type.

ContestCountingLogic (enum) N-of-M, Cumulative, Ranked order, or Straight party selection. (1-of-M is a special case of N-of-M.) The tabulation logic for a straight party selection Contest is implicitly 1-of-M, but with side-effects for other Contests.

NaturalNumber Integer greater than zero.

Text Character string.

WholeNumber Integer greater than or equal to zero.

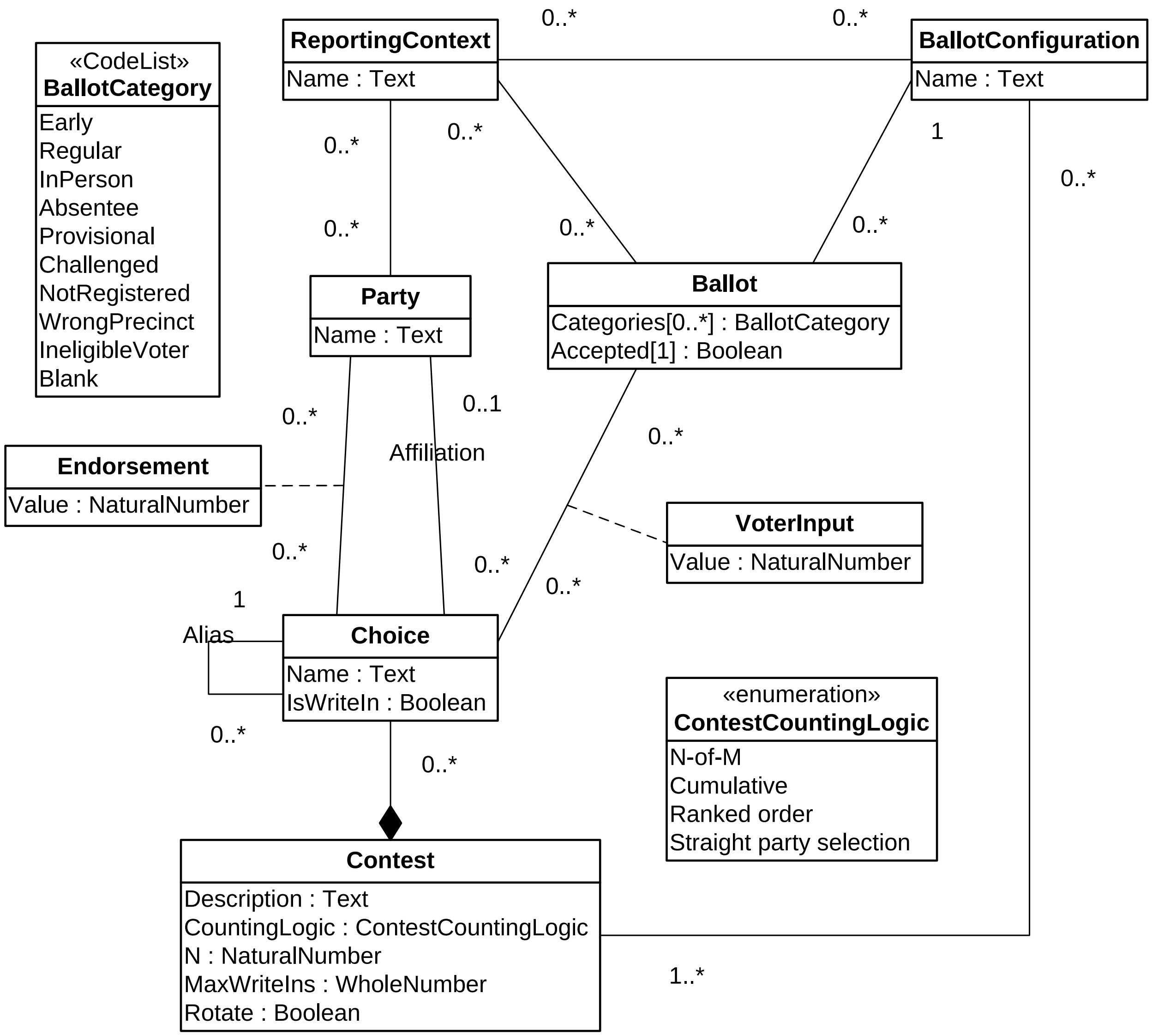
3.2 Classes

3.2.1 Ballot

The technology-independent, conceptual equivalent of a traditional paper ballot. The Contests that appear on a particular Ballot are defined by its BallotConfiguration. The applicable Reporting-Contexts include all those specified for its BallotConfiguration, but additional ReportingContexts may be specified for the individual Ballot.

Attributes of Ballot:

Figure 1: Data model for election definition and votes (Votetest)



Categories Arbitrary, jurisdiction-defined tags applied to the Ballot; e.g., Early, Regular, In-Person, Absentee, Provisional, Challenged, NotRegistered, WrongPrecinct, IneligibleVoter, Blank.

Accepted True if the Ballot should be counted, false if not (e.g., for a provisional Ballot that was not accepted).

3.2.2 BallotConfiguration

Set of Contests and ReportingContexts that is inherited by all Ballots of that configuration. BallotConfiguration is the conceptual equivalent of the paper ballot form that is handed to voters, while Ballot represents the filled-in ballot that goes in the ballot box. Depending on the type of election and local practices, a jurisdiction would define a separate BallotConfiguration for each precinct, each split within a precinct, and/or for each political party.

Attributes of BallotConfiguration:

Name Human-readable identifier.

A closely related concept that is defined in VVSG 2.0, but not included in the data model, is ballot style. The presentation issues that distinguish multiple ballot styles for the same ballot configuration from each other have been abstracted out of the model, and to an extent it is possible to merge the concepts with negligible impact. (In Votetest, this class was called BallotStyle simply because that term was more familiar to the audience than ballot configuration.) However, the distinction between the two concepts becomes important in the reporting requirements.

3.2.3 Choice

One of the things you can vote on in a Contest, such as a candidate, a political party, or yes or no. Choice is scoped by Contest, so even if the same person runs as a candidate in two or more Contests, those separate candidacies are represented by separate Choices. Choices do not map 1:1 with ballot positions—a Choice uniquely identifies a candidate, while a given ballot position might just be a generic write-in slot.

Attributes of Choice:

Name Human-readable identifier.

IsWriteIn True if the Choice is a write-in candidate, false if not.

N.B., Choices could have complex descriptive data associated with them that must be displayed to the user somehow, but this capability was not needed by Votetest.

3.2.4 Contest

Subdivision of a Ballot corresponding to a single question being put before the voters, consisting of header text, a discrete set of Choices, and possibly write-in opportunities. It is possible for a Contest to have zero Choices, e.g., if there are no registered candidates but write-ins are being accepted. Choices corresponding to the candidates written in would be added later.

Attributes of Contest:

Description Human-readable header text.

CountingLogic Identifies the tabulation method used for the Contest.

N For CountingLogic other than ranked order, N is the maximum number of votes that may be cast in the Contest by a given voter. In an N-of-M Contest, the voter may cast at most one vote for each Choice, so N is equal to the maximum number of Choices that the voter may select without overvoting.[1] In a cumulative Contest, there is no such constraint—the voter may cast more than one vote for a given Choice.

Typically, N also is the number of winners for the Contest, but not necessarily. The voting system only needs to gather votes and report the totals; the picking of winners may be an external process impacted by election law, late-breaking judicial rulings, etc. However, for ranked order Contests, N *is* specifically the number of Choices to be elected, and has no other meaning.

MaxWriteIns The number of ballot positions allocated for write-ins; the maximum number of candidates that the voter may write in. Any value between zero and N is possible.

Rotate True if the ordering of Choices within the Contest should be rotated, false if not.

3.2.5 Party

Surrogate for real-world political party.

Attributes of Party:

Name Unique human-readable identifier.

3.2.6 ReportingContext

Particular scope within which the system must be capable of generating reports. E.g., to support reporting at the precinct level, there must be a ReportingContext for each precinct.

The association between ReportingContexts and individual tabulators, precincts, election districts, political parties, ballot categories, or other arbitrary scopes of reporting is jurisdiction-defined and jurisdiction-managed, mostly using BallotConfigurations. Any relationship between particular administrative divisions (predicts, election districts, etc.) and Contests appearing on the ballot in those administrative divisions is implemented by BallotConfigurations.

[1] The value of M, for N-of-M voting, is simply the number of Choices associated with the Contest and is not explicitly modelled.

The ways in which ReportingContexts overlap or include one another is entirely determined by the assignment of multiple ReportingContexts to BallotConfigurations and Ballots.

Attributes of ReportingContext:

Name Human-readable identifier.

3.3 Named associations

3.3.1 Affiliation

Identifies the Party to which a candidate claims allegiance. Does not necessarily have anything to do with Endorsements.

3.3.2 Alias

Identifies an alternative Choice that for tabulation purposes is considered equivalent to a particular canonical Choice. Aliases will normally be variant spellings of a candidate's name that appeared in write-in positions.

3.3.3 Endorsement

Identifies a voter response that would be implied by a straight party vote for the endorsing Party. Does not necessarily have anything to do with Affiliation.

Attributes of Endorsement:

Value Analogous to VoterInput Value, this is the vote recommended by the endorser.

In a 1-of-M or N-of-M Contest, an Endorsement with Value = 1 would exist for the single Choice or for each of the Choices endorsed by the Party.

In a Cumulative Contest, Value may take on values greater than 1. For example, if the Party recommended that voters cast two votes for the first Choice and one vote for the second, an Endorsement with Value = 2 would exist for the first Choice and an Endorsement with Value = 1 would exist for the second Choice.

In a Ranked order Contest, Value contains the ranking that the Party recommends that voters assign to each Choice, with Value = 1 for the most preferred Choice.

3.3.4 VoterInput

The response that a particular Ballot provides for a particular Choice.

Attributes of VoterInput:

Value The response of the voter in some ballot position. The absence of a response is equivalent to a Value of 0 except in ranked order contests, where the behavior is implementation-defined.

In a 1-of-M or N-of-M Contest, a VoterInput with Value = 1 would exist for the single Choice or for each of the Choices for which the voter voted.

In a Cumulative Contest, Value may take on values greater than 1. For example, if a voter cast two votes for the first Choice and one vote for the second, a VoterInput with Value = 2 would exist for the first Choice and a VoterInput with Value = 1 would exist for the second Choice.

In a Ranked order Contest, Value contains the ranking that the voter assigns to each Choice, with Value = 1 for the most preferred Choice.

3.4 Unnamed associations

3.4.1 Ballot–BallotConfiguration

Every Ballot has a BallotConfiguration that determines which Contests appear on it as well as ReportingContexts in which it should be reported.

3.4.2 Ballot–ReportingContext

Every Ballot must be reported in at least one ReportingContext (per Constraint V), and will usually be reported in several (to implement multiple levels of reporting). In most cases, this follows as a consequence of BallotConfiguration–ReportingContext associations. However, in cases where the ReportingContexts in which a Ballot should be reported are not fully determined by its BallotConfiguration, a Ballot may be directly and explicitly associated with ReportingContexts.

3.4.3 BallotConfiguration–Contest

A BallotConfiguration identifies a set of Contests that appear on every Ballot having that configuration.

3.4.4 BallotConfiguration–ReportingContext

As described under Ballot–ReportingContext, the ReportingContexts in which a Ballot should be reported are usually determined by associations between its BallotConfiguration and ReportingContexts.

3.4.5 Choice–Contest

Every Choice belongs to exactly one Contest.

3.4.6 Party–ReportingContext

In primary elections, ReportingContexts may be established to enable breaking down results by Party even in non-party-specific Contests.

3.5 Constraints

I. For N-of-M and straight party selection Contests, the Value attribute of VoterInput or Endorsement must be 1. For cumulative Contests, $1 \leq \text{Value} \leq \text{N}$. (Deliberately, there is no analogous constraint for ranked order Contests.)

II. (In Contest) $\text{N} > 0$.

III. (In Contest) $0 \leq \text{MaxWriteIns} \leq \text{N}$.

IV. In Contests with CountingLogic = Straight party selection, N = 1 and MaxWriteIns = 0.

V. Every Ballot must be associated with at least one ReportingContext either directly or through its BallotConfiguration. (Otherwise the Ballot would never be reported.)

VI. A Ballot cannot have a VoterInput for a Choice in a Contest that does not appear in its BallotConfiguration.

VII. A given BallotConfiguration may contain at most one Contest with CountingLogic = Straight party selection.

VIII. A Contest with CountingLogic = Straight party selection cannot be straight-party-votable (i.e., there can be no Endorsements referring to its Choices).

IX. In Contests with CountingLogic = Straight party selection, the Names of the Choices must match the Names of Parties.

X. Party names must be unique.

XI. The Choice that an Alias cites as canonical cannot be aliased. (Corollary: There can be no cycles or self-referential Aliases.)

XII. The Choice that an Alias cites as canonical must be in the same Contest.

XIII. The Choice referenced by an Endorsement must be canonical (it cannot be an Alias).

XIV. A Ballot cannot have VoterInput for more write-in Choices in a given Contest than is allowed by the MaxWriteIns attribute of the Contest.

Votetest imposed two other constraints for testing purposes, but these are not appropriate for other applications of the model:

- A Ballot may not simultaneously have VoterInput for a Choice and an Alias of that Choice. (The handling of double votes for a given candidate resulting from write-in reconciliation is deliberately unspecified in the VVSG, so for testing purposes it was considered an error.)
- A Ballot may not simultaneously have VoterInput in a straight-party-votable Contest and a straight party vote that implies votes in that same Contest. (Resolution of straight party overrides is deliberately unspecified in the VVSG, so for testing purposes they were considered to be errors.)

3.6 Usage for all VVSG voting variations

3.6.1 In-person voting

No special requirements.

3.6.2 Absentee voting

Absentee voting is implemented in several different ways in practice, and it can be implemented in several different ways using this model.

1. Absentee Ballots can be tagged with the Absentee category and otherwise mingled with other Ballots.
2. A separate ReportingContext can be created for absentee Ballots and applied to the individual absentee Ballots.
3. A separate BallotConfiguration can be used for absentee Ballots.

While the first option is the least invasive, absentee Ballots are in practice sometimes processed as a separate precinct, which usually means both a separate ReportingContext and a separate BallotConfiguration.

3.6.3 Review-required ballots

Use Categories and Accepted attributes of Ballot as needed.

3.6.4 Write-ins

The number of write-ins permitted is an attribute of the Contest. If the write-in is new, a new Choice is created for it (with IsWriteIn = true). Votes are then associated with that Choice. Alias associations are created as applicable during write-in reconciliation.

3.6.5 Split precincts

Ballots are associated with the ReportingContexts pertaining to the applicable precinct and election district. If different BallotConfigurations are used for each split, the associations can be made on the BallotConfigurations. Otherwise, each Ballot must be individually associated.

3.6.6 Straight party voting

A single Contest is created with CountingLogic = Straight party selection and Choice Names being equal to the Names of the available Parties. In every other Contest that is straight-party-votable, the straight party behaviors are configured by creating Endorsement associations between the Choices and the Parties.

3.6.7 Cross-party endorsement

See straight party voting. Create additional Endorsement associations as needed for multiply endorsed Choices.

3.6.8 Ballot rotation

Rotate is a Boolean attribute of Contest. The implementation of variable mapping between Choices and ballot positions is out of scope because ballot positions are abstracted out of the model. However, in paper-based systems, rotation may involve a proliferation of ballot styles for each ballot configuration.

3.6.9 Primary elections

Create BallotConfigurations and ReportingContexts as needed to support the different political parties and unaffiliated voters. Non-party-specific Contests appear in all BallotConfigurations while party-specific Contests only appear in those BallotConfigurations applicable to the relevant Party.

3.6.10 Closed primaries

Assignment of BallotConfigurations to voters is procedural and out of scope.

3.6.11 Open primaries

Assignment of BallotConfigurations to voters is procedural and out of scope.

3.6.12 Provisional / challenged ballots

Use Categories and Accepted attributes of Ballot as needed.

3.6.13 1-of-M voting

Set ContestCountingLogic = N-of-M and set N = 1.

3.6.14 N-of-M voting

Set ContestCountingLogic = N-of-M and set N appropriately.

3.6.15 Cumulative voting

Set ContestCountingLogic = Cumulative and set N appropriately.

3.6.16 Ranked order voting

Set ContestCountingLogic = Ranked order and set N appropriately. VoterInput Values specify the rankings as provided on each Ballot.

4 Reporting

Figure 2 shows an example report that satisfies VVSG requirements for contests other than ranked order voting. Certain requirements are satisfied through indirect means. Part 1 Req. 7.8.3.2-C.1 (report separate ballot counts for each party in primary elections) is satisfied because each party gets a different ballot configuration and counts are already broken down by ballot configuration. Part 1 Req. 7.8.3.2-C.2 (report counted provisional ballots) and Req. 7.8.3.2-C.3 (report blank ballots) are satisfied by assuming support for the more general capability to break down ballot counts by category and defining categories for blank ballots and provisional ballots.

Figure 3 shows an extension of the Votetest data model to support the vote data reporting requirements of the VVSG as interpreted by the example. Note that other reports, such as equipment readiness reports and audit log reports, are not covered by this model.

4.1 Classes

The new classes added for vote data reporting functionality are a top-level class, VoteDataReport, and six classes that are related to it by composite aggregation. The six contained classes include four that give structure to Contest-independent ballot counts, which the VVSG requires both in total and broken down by BallotConfiguration, and two that give structure to the vote totals and ballot counts that must be reported for each Contest. The pertinent requirements appear in Part 1 Sections 7.8.3.2 and 7.8.3.3 of the VVSG.

Every part of the VoteDataReport is scoped by the ReportingContext with which it is associated and the Timestamp indicating the point in time at which the report was generated.

4.1.1 CategoryCounts

Report of the number of Ballots of a particular BallotCategory (but any BallotConfiguration) that were read and counted within the ReportingContext. The distinction between "read" and "counted" is as defined in the VVSG, with provisional and challenged ballots normally accounting for any ballots that were read but not counted.

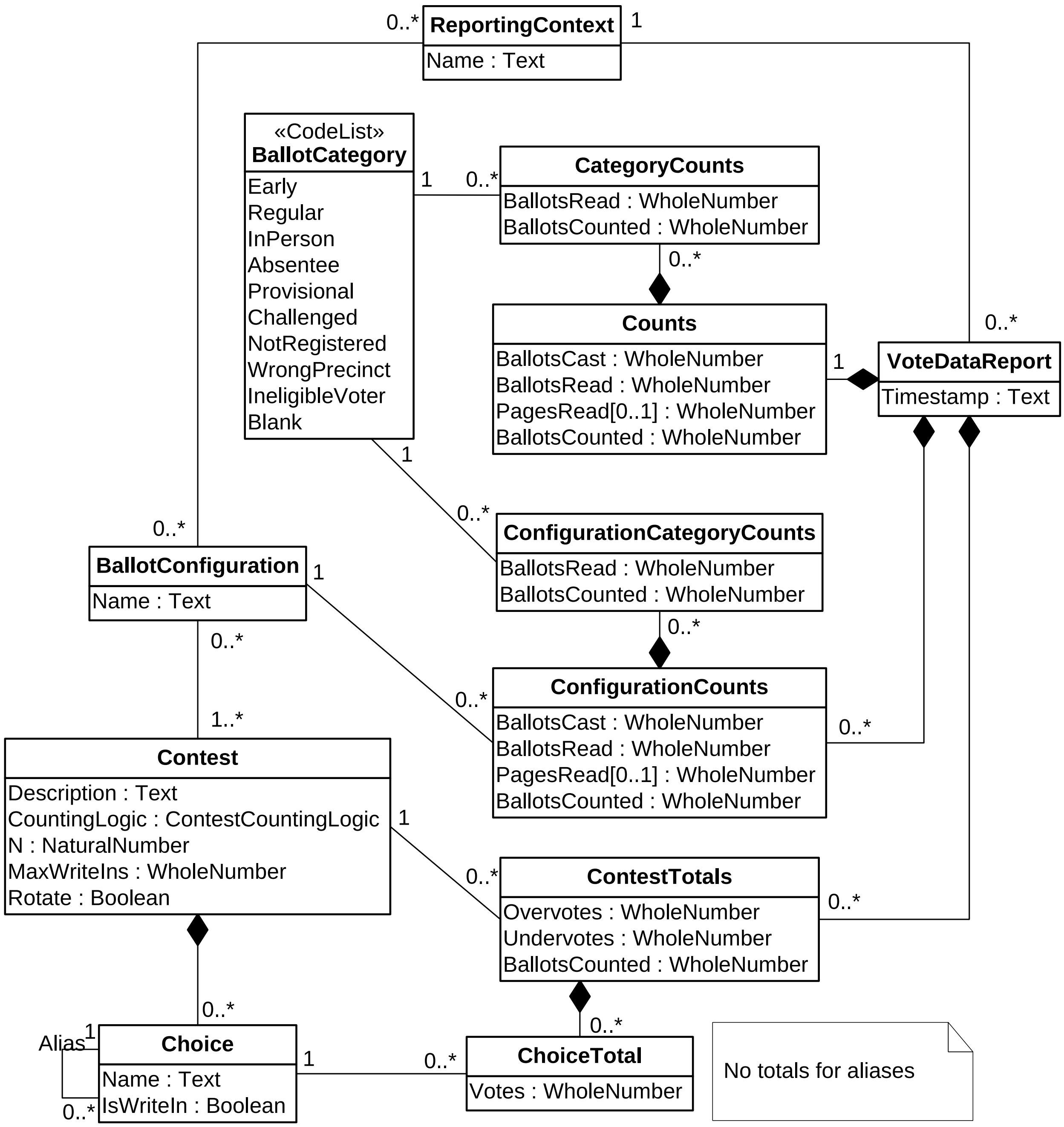
4.1.2 ChoiceTotal

Report of the number of valid votes for a particular Choice. Totals are reported only for canonical Choices; votes for Aliases are included in the total for the canonical Choice.

Figure 2: Sample report

```
Report for context Precinct 1 generated 2009-03-19 10:07:30-0400

                         BALLOT COUNTS

Configuration                              Cast    Read    Counted
-------------                              ----    ----    -------
Total                                        15      15         14
                        Provisional                   2          1
                              Blank                   1          1
Bipartisan Party Style                        5       5          5
                        Provisional                   1          1
                              Blank                   1          1
Moderate Party Style                         10      10          9
                        Provisional                   1          0

                          VOTE TOTALS

Nonpartisan office, vote for at most 1
Car Tay Fower                                        4
Tayra Tree                                           3
Overvotes                                            1
Undervotes                                           6
Counted ballots                                     14

Nominee of the Bipartisan Party, vote for at most 1
Beeso Tu                                             2
Oona Won                                             1
Overvotes                                            0
Undervotes                                           2
Counted ballots                                      5

Nominee of the Moderate Party, vote for at most 1
Wu Fife                                              5
Nada Zayro                                           0
Overvotes                                            0
Undervotes                                           4
Counted ballots                                      9
```

Figure 3: Data model for reporting



4.1.3 ConfigurationCategoryCounts

Report of the number of Ballots of a particular BallotCategory and BallotConfiguration that were read and counted within the ReportingContext. Similar to CategoryCounts, but scoped by a specific BallotConfiguration.

4.1.4 ConfigurationCounts

Report of the number of Ballots of a particular BallotConfiguration (but any BallotCategory) that were cast, read and counted within the ReportingContext. The distinction between "cast," "read" and "counted" is as defined in the VVSG, with unreadable paper ballots normally accounting for any ballots that were cast but not read. An optional PagesRead attribute is provided to enable satisfaction of VVSG Part 1 Req. 7.8.3.2-B.1, which only applies when there are multi-page paper ballots.

4.1.5 ContestTotals

Report of the number of overvotes and undervotes for a particular Contest, and the number of Ballots including that Contest that were counted within the ReportingContext.

4.1.6 Counts

Report of the number of Ballots of any BallotCategory and BallotConfiguration that were cast, read and counted within the ReportingContext. Similar to ConfigurationCounts, but not scoped by a specific BallotConfiguration.

In the trivial case of a VoteDataReport that reports on zero ballots, an instance of Counts is the only report content that would be required.

4.1.7 VoteDataReport

Report of all vote data pertinent to a particular ReportingContext as of the time indicated in the Timestamp attribute.

5 Design decisions

The limited size and scope of the data model presented here belie the subtlety of some aspects of its design. Some interesting design decisions are described in the following subsections.

5.1 No explicit associations among ReportingContexts

Election management systems generally provide some functionality to allow election officials to set up administrative divisions, such as precincts and election districts, in some regular structure that facilitates hierarchical accumulation of votes. The votes from a particular precinct are automatically included in the totals for the larger administrative divisions that contain it. The implied model is one of nested ReportingContexts forming a tree structure.

However, administrative divisions do not actually form a tree structure, as shown by the counterexample (and associated technical workarounds) of split precincts. Despite expectations to the contrary, the partitioning of the region that is done by one authority need have no similarity to the partitioning that is done by another authority. Any relationships among the administrative divisions at different levels are fortuitous and possibly convenient, but unreliable as a basis for roll-up of results.

Instead of representing an inheritance-based reporting structure with exceptions, which is then interpreted to determine the contexts in which a Ballot should be reported, the data model presented here associates all applicable ReportingContexts directly with BallotConfigurations (or, if needed, with individual Ballots). In concrete terms, instead of representing that Ballot B is in Precinct P and that Precinct P is in District D, it simply represents that Ballot B is in Precinct P and in District D. The fact that Precinct P is fully contained in District D, if it is the case, is evidenced by the fact that every ballot in Precinct P is also in District D; but that association between P and D is not directly represented in the model.

This approach supports any possible reporting structure without resort to workarounds. Moreover, it does not force any changes to the structural view that the election management system presents to election officials. The inheritance tree of ReportingContexts is simply "compiled" into an equivalent set of associations between BallotConfigurations and ReportingContexts. The preservation of the "syntactic sugar" view of ReportingContexts that an election management system presents may be desirable in a common data format, but it is not essential for the scope of functionality required by the VVSG.

5.2 VoterInput is an association class

At first glance, voter input is most likely viewed as an attribute of Ballot or as a regular class; however, refinement of the model reduced it to an association between Ballot and Choice whose attribute quantifies the number of votes cast in a non-ranked Contest or the ranking in a ranked order Contest. Conceptually, VoterInput establishes a new association between a Ballot and the Choices that is distinct from the association that already exists by virtue of the BallotConfiguration.

The Endorsement association is symmetrical with VoterInput except that it relates to a Party instead of a particular Ballot. As modelled, a party's set of endorsements forms a voting template, providing affiliated voters with a recommended set of votes to cast.

5.3 On the meaning of N

In the commonly used term N-of-M voting, there are three possible interpretations of N: the number of seats to fill (number of Choices to be elected), the maximum number of Choices that may be selected in the Contest by a given voter, *or* the maximum number of votes that may be cast in the Contest by a given voter. Usually, these quantities will be equal, but they are not *necessarily* equal, so it is important to decide which of them must be represented in a minimal data model.

In N-of-M Contests, Constraint I causes the latter two possibilities to be equivalent. Only in cumulative Contests might they be distinguished. However, the VVSG does not require support for esoteric variants of cumulative voting, so the latter two possibilities can be merged.

The output of the voting system is a list of choices with their vote totals, sorted into descending order. In order to generate this ranking, the voting system needs to know the maximum number of votes that may be cast in the Contest by a given voter, as this is what distinguishes votes from overvotes. However, the voting system does *not* need to know the number of seats to fill. Having generated a ranking of candidates, the voting system can leave the task of declaring winners as an exercise for election officials. Any complications resulting from the disqualification or death of candidates or other external factors then have no effect on the validity of the system's report. The vote totals have been reported and the task of the voting system is complete.

Unfortunately, this design decision cannot be applied consistently for ranked order voting. Since the voter is just ranking all of the candidates, there is no analog to maximum number of votes; however, an implementation of ranked order voting cannot reach a reportable result without knowing the number of winners. So in the case of ranked order voting, N is defined (inconsistently) as the number of Choices to be elected.

While it was expedient within the original application of the data model, the inconsistent interpretation of N for ranked order voting is a conceptual integrity compromise that should be removed through subclassing of Contest when support for ranked order voting is improved overall.

5.4 A Choice is scoped by a Contest

While a particular person may be running as a candidate in more than one Contest, that fact and the additional complexities attached to it are unnecessary for the function that the voting system must accomplish. Instead, we use a simpler abstraction that suffices for the functional view. The "black diamond" composite aggregation relationship from Choice into Contest specifies that a given Choice exists as a part of exactly one Contest. If a person runs as a candidate in two contests, then there are two separate Choices, one associated with each of the two contests, that happen to display the same name.

5.5 Accounting for every ballot

While it is commonly understood that some ballots may be cast that are not counted for one reason or another, the VVSG distinguishes that in paper-based systems there are actually three possible outcomes. A paper ballot that is found in a ballot box or received as a mail-in absentee ballot, but that is so damaged that its content cannot be recovered, is cast but not read or counted. A readable ballot of any sort that is not counted because of failure to register, voting in the wrong precinct, or other anomalies is cast and read but not counted. A ballot with no such problems is cast, read and counted. By separating the cases, it becomes possible to do ballot accounting in terms of physical ballots received *and* in terms of ballots interpreted by the voting system, without ambiguity.

5.6 Accounting for every vote

The content of ContestTotals, together with the associated ChoiceTotals, suffices to account for every vote cast in a given Contest and ReportingContext.

Letting V_i represent the value of the Votes attribute of ChoiceTotal associated with a particular Choice in the Contest, the following equation should hold:

$$\sum V_i + Overvotes + Undervotes = BallotsCounted \times N$$

Critically, the Overvotes and Undervotes values are defined in the VVSG as reporting the number of votes, as opposed to simply the number of Ballots that exhibited the relevant voting behavior, as is sometimes done.

6 Conclusion

We have presented a small data model that suffices to cover the election definition and vote reporting requirements of VVSG 2.0, with the exception of reporting for ranked order contests. Assuming that VVSG compliance is among the goals of a common data format, this contribution should expedite the process of constructing an underlying common data model, either by providing the starting point for that work or by inspiring others to propose significantly different alternatives.

Continued evolution of this model should include revisiting ranked order voting in coordination with an expansion of the ranked order reporting requirements in the VVSG. Depending on the goals and scope of a common data format, extensions to cover other reports such as equipment readiness reports and audit log reports may also be needed. It may also be desirable to represent explicitly the relationships that exist among ReportingContexts at different levels. Finally, additional attributes, such as for storing complex descriptive data associated with Choices, may need to be added to support important but non-standardized requirements of elections in practice.

Acknowledgment

Thanks to Ed Barkmeyer, Vadim Okun and Lynne Rosenthal for their reviews and comments.

References

[1] Election Assistance Commission. *Voluntary Voting System Guidelines Recommendations to the Election Assistance Commission*, August 31, 2007. `http://purl.org/net/dflater/VVSG/20070831`.

[2] Votetest test suite for the VVSG-NI, version 1.0, April 1, 2009. `http://vote.nist.gov/SystemTesting/reviewer-notes-votetest.htm`.

[3] OMG Unified Modeling Language specification, version 2.2. Documents formal/2009-02-02 and formal/2009-02-04, Object Management Group, February 2009. `http://www.omg.org/spec/UML/2.2/`.