Core Requirements and Testing

David Flater

Alan Goldfine

Today's agenda

1. Standards architecture (profiles, compliance points, formalized implementation statements…)
2. Software integrity and coding conventions
3. Methods for conformity assessment
   - Logic verification
   - Test protocols
4. Casting, counting, and reporting requirements
5. Process model
6. Performance and workmanship requirements
7. Issue paper
8. Research papers on VVSG maintenance and information sharing
9. Future work

Prior CRT deliverables to TGDC

- CRT changes in VVSG 1:
  - Revised glossary
  - Basic conformance clause
  - Modified MTBF test
- Deferred
  - Noncritical changes to 2002 VSS — including most CRT-specific work
  - Other items not doable by May 2005 deadline

Agenda

1. **Standards architecture** (profiles, compliance points, formalized implementation statements…)
2. Software integrity and coding conventions
3. Methods for conformity assessment
   - Logic verification
   - Test protocols
4. Casting, counting, and reporting requirements
5. Process model
6. Performance and workmanship requirements
7. Issue paper
8. Research papers on VVSG maintenance and information sharing
9. Future work

Usage of “standard”

- **Standard**: Document, established by consensus and approved by a recognized body, that provides, for common and repeated use, rules, guidelines or characteristics for activities or their results, aimed at the achievement of the optimum degree of order in a given context. (Glossary / ISO Guide 2-2004)
- **Not** necessarily a regulation
- Can be “voluntary guidelines”

Pieces of standards architecture

- Expanded conformance clause, with definitions of profiles
- Compliance points
- Formal implementation statement, citing profiles

Motivation

- TGDC Resolution #24-05, “Conformance clause;” #25-05, “Precise and Testable Requirements”
- Prerequisite for improved precision, testability, and traceability

Profiles

- Profile: specialization of a standard for a particular context, with constraints and extensions that are specific to that context. (Glossary def. 2)
- Formalize profiles for "categories"
- Add profiles for supported voting variations and optional functions
- Types of independent dual verification
- Defined in conformance clause (Sec. 4.2)
- Extensible (e.g., this state's profile)

Compliance points

- Identified, testable requirements
- Getting there
  - Extricate compound requirements from one another, make separate compliance points
  - Clarify general requirements via sub-requirements that are profile- and/or activity-specific
  - Refactor repeating, overlapping requirements

Implementation statement

- Vendor identifies profiles to which the system is believed to conform
- Applicable test cases are identified by profiles
- Certification is to those profiles only

Issues

- Sorting out will take time
- Identification, referencing and indexing of compliance points require a robust document production system
- Versioning of standard

Non-issues

- Conformance levels

Technical Guidelines Development Committee September 29, 2005 Plenary Meeting

Agenda

1. **Standards architecture** (**profiles**, compliance points, formalized implementation statements…) **(section 4.2.2)**
2. Software integrity and coding conventions
3. Methods for conformity assessment
   – Logic verification
   – Test protocols
4. Casting, counting, and reporting requirements
5. Process model
6. Performance and workmanship requirements
7. Issue paper
8. Research papers on VVSG maintenance and information sharing
9. Future work

Definition

Profile: (1) Subset of a standard for a particular constituency that identifies the features, options, parameters, and implementation requirements necessary for meeting a particular set of requirements. (2) Specialization of a standard for a particular context, with constraints and extensions that are specific to that context.

(1) ISO 8632 (Computer Graphics Metafile)

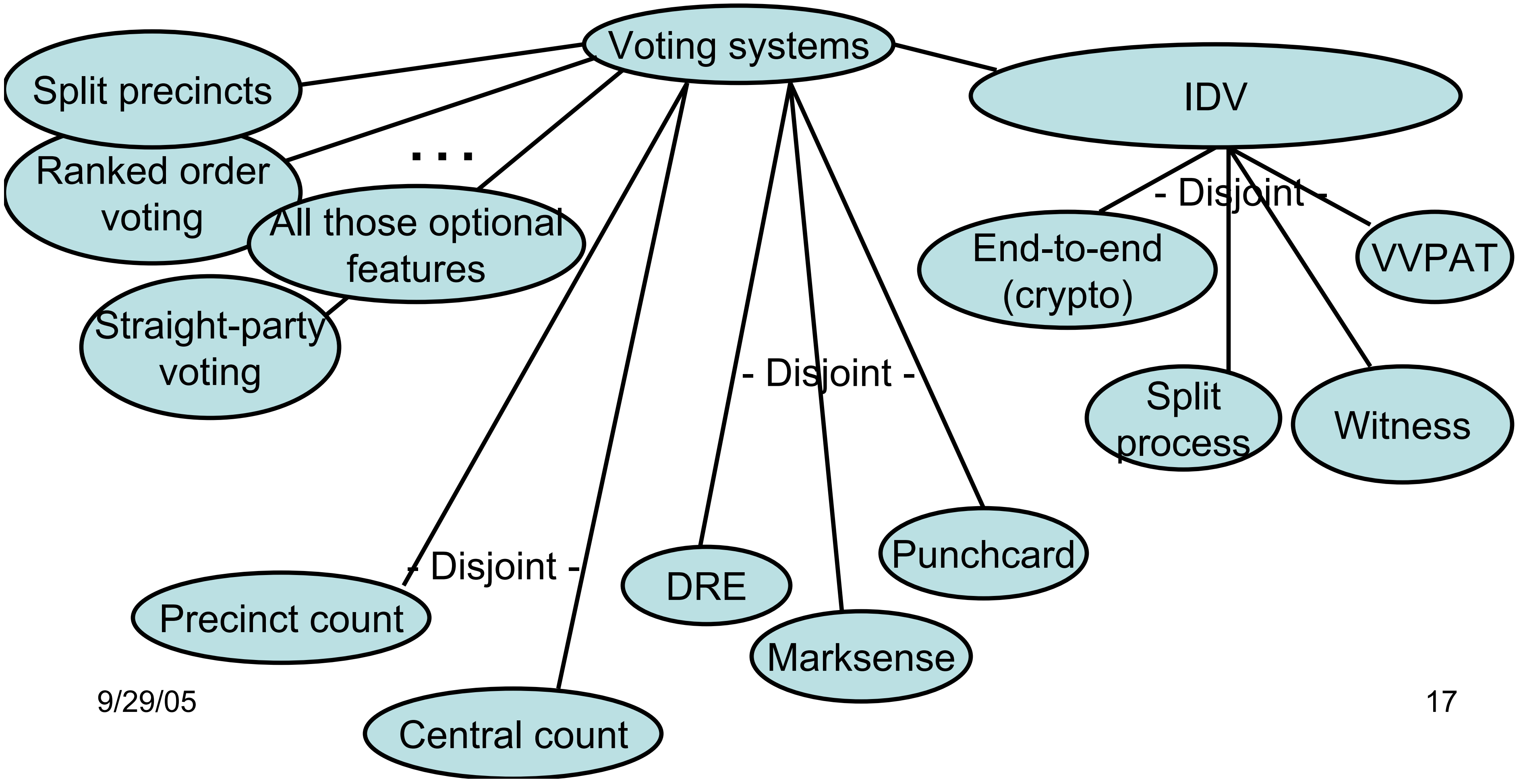
Profiles in implementation statement

- An implementation statement shall identify:
  - Exactly 1 profile from group 1
  - Exactly 1 profile from group 2
  - All applicable profiles from group 3
- Profile group 1: product classes, taxonomy ‘A’
  - Precinct count
  - Central count

Structure of profiles

- Profiles form a classification hierarchy
- Profiles may subsume other profiles (e.g., paper-based subsumes marksense and punchcard, and “all systems” subsumes everything)
- Profiles may be declared to be mutually exclusive, or not, as appropriate (e.g., paper-based and electronic overlap in the case of marksense)
- Constraint: in all cases, that which conforms to a subprofile also conforms to the subsuming profile
- Anything conforming to a profile of a standard conforms to that standard

The hierarchy so far



Profiles in compliance points

- Restricts applicability of compliance point to a precisely specified subset of voting systems
- Requirement 4.4.3.5-1.2 Systems conforming to the *DRE* profile shall maintain an accurate Cast Vote Record of each ballot cast.
- Requirement 4.4.2-2.13.1 In systems conforming to the *Provisional / challenged ballots* and *DRE* profiles, the DRE shall provide the capability to categorize each provisional/challenged ballot.

Profiles in conformity assessment

- Like compliance points, test cases and expert reviews are associated with specific profiles or combinations of profiles
- Only those conformity assessment activities indicated for the profiles claimed in the implementation statement are applicable

Profiles and traceability

- Using the profiles mechanism, jurisdictions may formally define their own specializations of the standard, provided that they do not conflict with the standard (that which conforms to a subprofile *must* conform to the subsuming profile)
- Conformance to the jurisdiction's profile is well-defined because the conformance clause is included by reference, as are all of the applicable compliance points

Technical Guidelines Development Committee
September 29, 2005 Plenary Meeting

Agenda

1. Standards architecture (profiles, compliance points, formalized implementation statements…)
2. **Software integrity and coding conventions (sections 4.3.1.1 and 4.3.4.1.1)**
3. Methods for conformity assessment
   - Logic verification
   - Test protocols
4. Casting, counting, and reporting requirements
5. Process model
6. Performance and workmanship requirements
7. Issue paper
8. Research papers on VVSG maintenance and information sharing
9. Future work

What they are

- Mostly, requirements on the *form* (not *function*) of source code
- Some requirements affecting software integrity, implemented as defensive coding practices
  - Error checking
  - Exception handling
  - Prohibit practices that are known risk factors for latent software faults and unverifiable code
  - Unresolved overlap with STS….

Motivation

- Started in 1990 VSS, expanded in 2002, expanded more in IEEE P1583 5.3.2b
- TGDC Resolution #29-05, "Ensuring Correctness of Software Code" (part 2)
- Enhance workmanship, security, integrity, testability, and maintainability of applications

2002 VSS / VVSG 1 status

- Mixture of mandatory and optional
- Vendors may substitute “published, reviewed, and industry-accepted coding conventions”
- Incorporated conventions have suffered from rapid obsolescence and limited applicability
- Some mandatory requirements had unintended consequences

Recommended changes

- Expand coding conventions addressing software integrity
  - Start with IEEE requirements
  - Make defensive coding requirements (error and range checking) more explicit (Sec. 4.3.1.1.2)
  - Require structured exception handling
- Clarify length limits (modules vs. callable units)
- Delete the rest; require use of “published, credible” coding conventions

Credible ≈ industry-accepted

- Coding conventions shall be considered credible if at least two different organizations with no ties to the creator of the rules or to the vendor seeking qualification independently decided to adopt them and made active use of them at some time within the three years before qualification was first sought.

Issues (1)

- Definition of “credible” is problematic
- General: prescriptions for how to write code versus open-ended expert review
  - Performance requirement “shall have high integrity” is too vague, not measurable
  - 2002 VSS, P1583 5.3.2b include prescriptions
  - STS favors open-ended expert review
  - Compromise: conservative prescriptions followed by STS review

Issues (2)

- Public comment, April 14: “The NASED Technical Committee has previously ruled that assembler code is permitted as long as the code meets all other requirements.” TBD – need to get a copy of that ruling, determine if it covers tabulation code, and if so, why.
- C doesn’t have structured exception handling
- Delete integrity requirements, etc. if “published, credible” replacements are found

Non-issues

- Assembly language in “hardware-related segments” and operating system software
- Grandfathering of stable code — part of general grandfathering strategy
- COTS or “slightly modified” COTS — part of COTS strategy, driven by security requirements, T.B.D.

Agenda

1. Standards architecture (profiles, compliance points, formalized implementation statements…)
2. Software integrity and coding conventions
3. **Methods for conformity assessment**
   - Logic verification
   - Test protocols
4. Casting, counting, and reporting requirements
5. Process model
6. Performance and workmanship requirements
7. Issue paper
8. Research papers on VVSG maintenance and information sharing
9. Future work

Conformity assessment overview

- Reviews
  - ***Logic verification***
  - Open-ended security review
  - Accessibility, usability reviews
  - Design requirement verification
- Tests
  - ***Test protocols***
  - Performance-based usability testing
  - Environmental tests

Agenda

1. Standards architecture (profiles, compliance points, formalized implementation statements…)
2. Software integrity and coding conventions
3. **Methods for conformity assessment**
   – **Logic verification (sections 4.5.2, 5.1.2, 5.3, & 6.3.1)**
   – Test protocols
4. Casting, counting, and reporting requirements
5. Process model
6. Performance and workmanship requirements
7. Issue paper
8. Research papers on VVSG maintenance and information sharing
9. Future work

What it is

- Formal characterization of software behavior within a carefully restricted scope
- Proof that this behavior conforms to specified assertions (i.e., votes are reported correctly in all cases)
- Complements [falsification] testing

Motivation

- TGDC Resolution #29-05, “Ensuring Correctness of Software Code”
- Higher level of assurance than functional testing alone
- Clarify objectives of source code review

How it works

- Vendor specifies pre- and post-conditions for each callable unit
- Vendor proves assertions regarding tabulation correctness
- Testing authority reviews, checks the math, and issues findings
  - Pre- and post-conditions correctly characterize the software
  - The assertions are satisfied

Issues

- Training required
- Limited scope = limited assurance; unlimited scope = impracticable
- Overlaps with security reviews

Non-issues

- Rice’s theorem

Agenda

1. Standards architecture (profiles, compliance points, formalized implementation statements…)
2. Software integrity and coding conventions
3. **Methods for conformity assessment**
   - Logic verification
   - **Test protocols (section 6.4)**
4. Casting, counting, and reporting requirements
5. Process model
6. Performance and workmanship requirements
7. Issue paper
8. Research papers on VVSG maintenance and information sharing
9. Future work

What it is

- General test template
- General pass criteria
- Collection of testing scenarios with implementation-specific behavior parameterized or abstracted out
- Functional tests, typical case tests, capacity tests, error case tests
- Performance-based usability tests might be integrated, T.B.D. (Template may differ)

Motivation

- TGDC Resolution #25-05, item 4 (test methods)
- TGDC Resolution #26-05, “Uniform Testing Methods and Procedures”
- Improved reproducibility

Changes from current

- Augments implementation-dependent, white box structural and functional testing
- Raises the baseline level of testing that all systems must pass
- Error rate, MTBF estimated from statistics collected across run of entire test suite

Issues

- Implementation-dependent white box testing: poor reproducibility, but hardly redundant
- Testing combinations of features requires extensive test suite
- Typical case tests – need input on what is typical

Today's agenda

1. Standards architecture (profiles, compliance points, formalized implementation statements…)
2. Software integrity and coding conventions
3. Methods for conformity assessment
   - Logic verification
   - Test protocols
4. **Casting, counting, and reporting requirements (sections 4.4.2 and 4.4.3)**
5. Process model
6. Performance and workmanship requirements
7. Issue paper
8. Research papers on VVSG maintenance and information sharing
9. Future work

Origins

- Mostly derived from 2002 VSS
- Refactored requirements to clarify and reduce redundancy
- Separated election administration concerns (best practices, Sec. 4.6)
- Added precision

Motivation

- TGDC Resolution #25-05, “Precise and Testable Requirements”

Significant changes

- In casting and counting, specified functional requirements for the many voting variations (optional profiles)
- In reporting, significantly revised requirements on the content of reports
- Accuracy of reported totals now defined by logic model

Reporting issues

- Cast, *read*, and counted: three concepts, not two
- Reporting levels: tabulator, precinct, election district, and jurisdiction (state)
  - 2002 VSS unclear on what levels are required
  - Generic facility to define arbitrary reporting contexts is not required
- Manual / optional processing of ballots with write-in votes: outside the system, unverifiable. Can these possibly conform to the write-ins profile?
- Define "unofficial"

Agenda

1. Standards architecture (profiles, compliance points, formalized implementation statements…)
2. Software integrity and coding conventions
3. Methods for conformity assessment
   - Logic verification
   - Test protocols
4. Casting, counting, and reporting requirements
5. **Process model (section 4.5.1)**
6. Performance and workmanship requirements
7. Issue paper
8. Research papers on VVSG maintenance and information sharing
9. Future work

What it is

- A formal model of the elections process in both graphical and textual representations
- An identification of the dependencies among activities and objects
- Informative, not normative

Motivation

- TGDC Resolution #33-05, “Glossary and Voting Model”
- Complements glossary to further clarify vocabulary
- Helps to specify intended scope of requirements

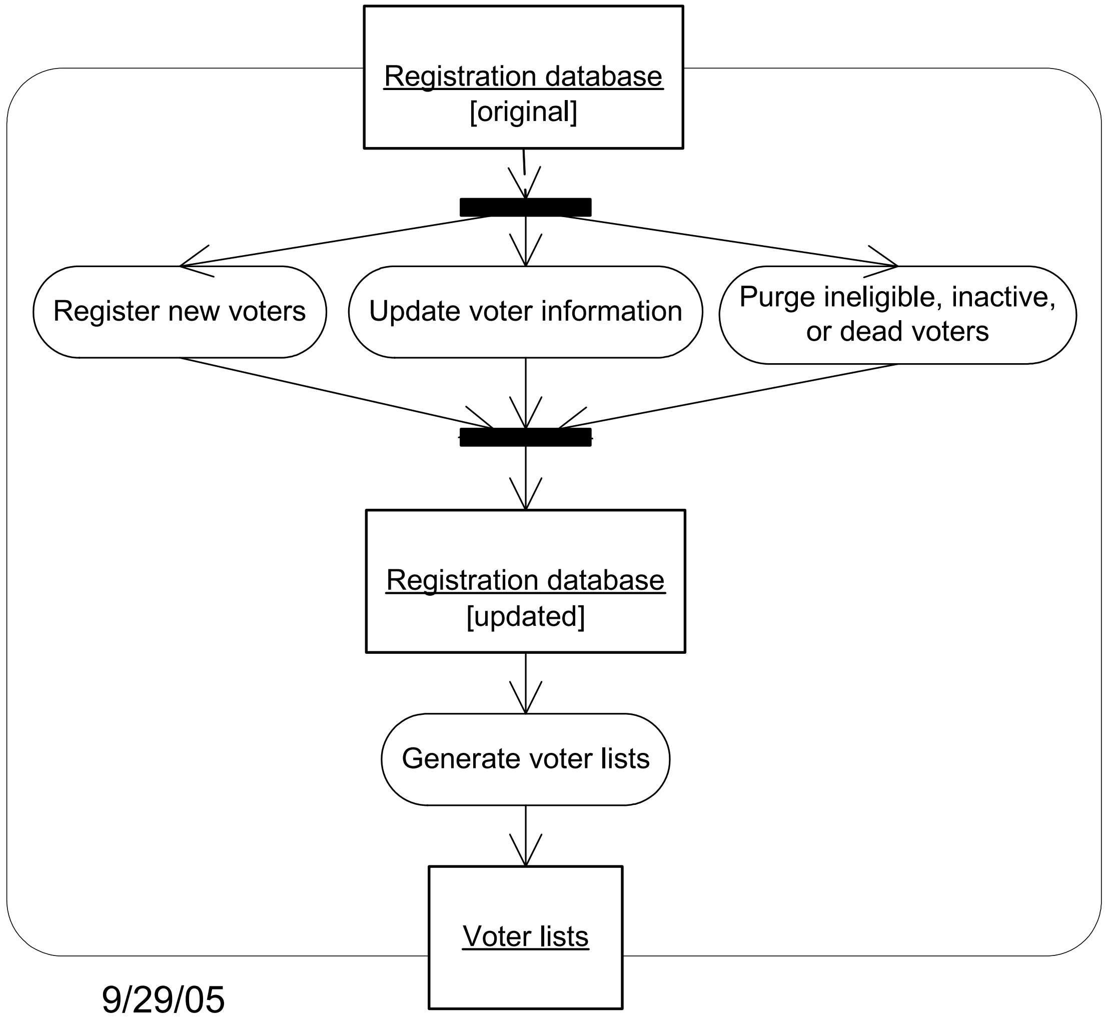
Origins

- Review of previous work
- Input from CRT subcommittee
- Reconciliation of conflicting vocabulary and models
- Elaboration as needed

Language

- Graphical representation is activity diagram as defined in Unified Modeling Language version 2.0
- Textual representation is based on Petri Net Linear Form

Technical Guidelines Development Committee September 29, 2005 Plenary Meeting



Registration database [original]
Register new voters
Update voter information
Purge ineligible, inactive, or dead voters
Registration database [updated]
Generate voter lists
Voter lists

Diagram: Register voters

Input: [Registration database [original]]
Output: [Voter lists]

```
[Registration database [original]]
  -><ForkNode>{
    ->(Register new voters)
      -><JoinNode *j>,
    ->(Update voter information)
      -><*j>,
    ->(Purge ineligible, inactive, or dead voters)
      -><*j>
  };
<*j>
  ->[Registration database [updated]]
  ->(Generate voter lists)
  ->[Voter lists].
```

Issues

- Vocabulary is still evolving
- Probably never match any state’s processes perfectly