

The issue ...



“Let’s see how my vote is counted”

©Automatisering Gids 2003.

Counting Votes with Formal Methods (p.1 of 40)

Counting Votes with Formal Methods

Bart Jacobs

with Engelbert Hubbers, Joseph Kiniry, Martijn Oostdijk

bart@cs.kun.nl

<http://www.cs.kun.nl/~bart>.

Security of Systems (SoS),
Department of Computer Science
University of Nijmegen, NL

Invited talk, AMAST, Stirling, 16 july 2004

Counting Votes with Formal Methods (p.2 of 40)

Contents

- I. Voting issues
- II. Experimental “remote voting” system in NL
- III. Vote counting software & formal methods
- IV. Experiences
- V. Concluding remarks

I. Voting issues

Counting Votes with Formal Methods (p.3 of 40)

Counting Votes with Formal Methods (p.4 of 40)

Topic

Concretely

- European Elections of June 2004 allowed “Remote Voting” or “*Kiezen op Afstand*” (KOA) in the Netherlands via *internet* and *phone*
- Intended for expatriats, after registration

Abstractly

- New technology requires reconsideration of the essentials of voting (Similarly with electronic signatures, for instance)
- Risk identification

Counting Votes with Formal Methods (p.5 of 40)

Own involvement in remote-voting

- Professional interest in challenging & hot topic
- Writer of patent application in this area (september 2002, in EC & US)
- Member of *expert panel* at ministry (august 2003)
- Non-commercial audit of webservice in voting trial (november 2003)
- Commercial assignment to build vote counting software (march 2004)

But: no detailed knowledge of entire system!

Counting Votes with Formal Methods (p.6 of 40)

Requirements for voting

Correctness

- Each valid vote should contribute precisely once to the final outcome.
- The result should be verifiable: recounts possible

Security

- **Confidentiality:** votes should not be traceable—to prevent use of force and sale of votes
- **Integrity:** the expressed vote should contribute (unchanged) to the outcome
- **Availability:** vote intentions should be realisable

Counting Votes with Formal Methods (p.7 of 40)

Different approaches

1. Traditional paper ballots
2. Voting machines (in voting stations)
3. Online voting systems

Counting Votes with Formal Methods (p.8 of 40)

Paper ballots

Advantages

- Low-tech, transparent system
- Security lies in distributed character: large scale tampering is difficult, and easy to observe
- Vote counting happens in public
- Recount possible with original, unprocessed data

Disadvantages

- No automatic processing: labour-intensive and slow
- Vote expressions may be ambiguous (Florida 2000)
- Voters need to travel (and be around)
- Software higher up in the processing chain?

Counting Votes with Formal Methods (p.9 of 40)

Voting Machines



- Widely used in NL (since 1991) & IRL
- Main supplier: Nedap
- Dedicated system (25,000 loc)
- No indication of irregularities
- Internal mechanics is secret
- Independent evaluation is required (done by TNO)
- Evaluation reports are secret

Counting Votes with Formal Methods (p.10 of 40)

Voting Machines, continued

Advantages

- automatic processing of results: efficient and fast
- vote expression is unambiguous

Disadvantages

- “Processing gap” between expression and recording of the vote.
- Recount only possible on already processed votes
- Voter cannot verify that the vote is registered correctly (but correctness of totals can be checked)
- Voters need to travel

Counting Votes with Formal Methods (p.11 of 40)

US Discussion on paper trails

- Much controversy, after failing evaluations and bad (leaked) code
- Suggested solution: voting machine prints a paper when the voter has finished
- The voter inspects the printout, and if it is correct, the paper is deposited in a special box
- The machine provides a preliminary total; the ballots are used for a recount, if requested.
- Around 1000 computing professionals have signed a petition to include such a voter-verifiable audit trail.
- Probably it will be demanded by US federal law
- But: with trails, mechanical failures more likely.

Counting Votes with Formal Methods (p.12 of 40)

Voting machines & trust

“Consequently, the integrity of elections rests on blind faith in the vendors, their employees, inspection laboratories, and people who may have access—legitimate or illegitimate—to the machine software”

“Democracy should not depend on blind faith”

(From: Dill, Schneier and Simons, *Voting and Technology: Who Gets to Count Your Vote?*, Communications of the ACM, August 2003)

Counting Votes with Formal Methods (p.13 of 40)

Voting machines, looking back

- The (gradual) introduction of these voting machines in NL since 1991 was uncontroversial
- Openness (of software) was not an issue at the time.
- Currently controversy in IRL, and questions in NL too
- Simply requiring **open source** is not the (complete) answer:
 - Is the published code actually running?
 - What about OS, compilers, hardware, ... ?
 - Open source requires different business model.
- Reliance on evaluators raises questions: can they
 - handle the complexity
 - be really trusted?

Counting Votes with Formal Methods (p.14 of 40)

Next step ... online voting systems

- **Main advantage:** voters don't need to travel, or be in NL. This may increase participation.
- **Main disadvantage:** security risks.
 - online systems are accessible by hackers
 - centralisation increases vulnerability
 - individual freedom to vote is not guaranteed at home
 - also processing gap between expression and registration of votes.

Counting Votes with Formal Methods (p.15 of 40)

II. Remote Voting System (KOA) in NL

Counting Votes with Formal Methods (p.16 of 40)

Background

- Open bidding won by LogicaCMG, to set up a voting *service* (early 2003)
- Experiments and evaluations, notably by third parties, in second half of 2003
- Limited, one-time, low-tech experiment modeled after voting by (ordinary) mail:
 - Explicit registration with user-defined access code (as PIN, or password)
 - Confidentiality & integrity not guaranteed at home
- Phone as alternative for internet

Counting Votes with Formal Methods (p.17 of 40)

Codes, codes, codes, ...

Code	Distribution	Function
voter code (<i>stemcode</i>)	after registration	identification when voting
access code (<i>toegangscode</i>)	at registration (self-chosen)	authentication when voting
candidate code (<i>kandidaatcode</i>)	in ballot (see next slide)	choice
transaction code (<i>transactiecode</i>)	after voting	participation check after election

Counting Votes with Formal Methods (p.18 of 40)

Ballot form, one for each voter

Overzicht van Kandidaten Gebruikersproef oktober 2003

1 Europese Kleurenpartij (EKP)	2 Planten voor het Volk (P.v.h.V.)	3 EUROPESE WEERMAN- NEN	4	5 Europese Dierall (EDA)
1. Azuur, W.F. (Walter) (m) %Gravenhage 192508708	1. Roos, G. (Gerard) (m) Reuver% 891056162	1. Wolk, E Hekman% 273820126	1. Vilt, M.W.P. (Marcel) (m) %Gravenhage 717032568	1. de Olifant, K.L. (Klaas) Arends% 147127760
2. de Parelgrijs, C. (Cort) (m) Brouwer% 818495260	2. van Chryasant, C.J. (Christiaan) (m) Brouwer% 738683929	2. de Sneeuw, C.C. %Gravenhage 522715084	2. van Zijden, Y.M. (Yvonne) (v) Utrecht% 317900602	2. Örka, W. (Walter) Arends% 550075042
3. Blauw, Y.M. (Yvonne) (m) Wolfs% 872093445	3. Tulp, V. (Volk) (v) Buda 555146248	3. van der Kou, H.K.L. Franker% 445189925	3. Velours, M.L. (Marjol) (v) Brouwer% 570532966	3. Tijger, L.R. (Luis) Santenberg% 304856204
4. Kersen-rood, G.M.H. (Gerrit) (v) %Gravenhage 865884403	4. Lelle, S.A. (Sander) (m) %Gravenhage 856296002	4. Hagel, G.F. Loocht% 779276723	4. Linnen, G.A. (Gabriel) (m) Brouwer% 247213421	4. Leguaan, E. (Erik) %Gravenhage 709233306

Counting Votes with Formal Methods (p.19 of 40)

What happens in internetvoting

1. Voter uses webbrowser to contact `www.internetstembureau.nl`, and establishes a secure SSL connection
2. Identification & authentication, resulting in check
3. Vote is cast, after confirmation of choice
4. Vote is recorded, in encrypted form
5. Voter gets transaction code, as confirmation of recording
6. At the end, encrypted database of votes is extracted
7. Decryption by head of voting station, and **counting!**
8. Recount, if needed, on basis of *same database* of processed votes.

Counting Votes with Formal Methods (p.20 of 40)

Some security issues & questions

- Protection of webserver (intrusion, denial of service)
- Protection of communication: voter must check SSL-certificate in padlock
- Influence of system administrators: procedural measures
- Recording of “raw data” from webserver for recount?
 - Blinding is necessary for confidentiality (no IP addresses visible!): “logging tension”
- Personalised ballots give better protection?
 - More confusion likely
 - People may feel being watched

Counting Votes with Formal Methods (p.21 of 40)

Internetvoting: results

Dutch expatriates	600,000
Typical participation	20,000 – 30,000
Registration for EU'04	14,000
Preference for E-voting	7,000
E-votes cast	5,000

- Outcome: E-votes a bit more rightwing
- No mayor attacks detected (during 7 day 24 hour operation & monitoring)

Counting Votes with Formal Methods (p.22 of 40)

III. Counting software & formal methods

Tally system “KOA”: background

- Original system for webserver, vote collection & counting by LogicaCMG.
- Vote counting was split off, in separate (commercial) bidding
- Won bij Nijmegen Univ. with bid including use of formal methods (JML & ESC/Java2)
- Main challenge: not formal methods, but strict time schedule: about 4 weeks, 3 developers (with PhDs)
- Hence: only lightweight specification, with best-effort verification:
 - no theorem proving (using e.g. LOOP tool)
 - static checking & testing

Counting Votes with Formal Methods (p.23 of 40)

Counting Votes with Formal Methods (p.24 of 40)

JML: Java Modeling Language

In *JML* [Leavens et al.] one may add specifications as special comments in Java code, for:

- Class invariants and constraints
- Method specifications:

```
/*@ behavior
@ requires <precondition>
@ modifiable <items that may be modified>
@ diverges <precondition for non-termination>
@ ensures <postcond for normal termination>
@ signals <postcond for exceptional
@ termination>
@*/
void method() { ... }
```

- Pure methods & model variables (specification only)

Counting Votes with Formal Methods (p.25 of 40)

JML: example

JML method specifications may clarify the behaviour of Java methods:

```
/*@ normal_behavior
@ requires x >= 0;
@ modifiable \nothing;
@ ensures \result * \result <= x &&
@ x < (\result + 1) * (\result + 1)
@*/
int f(int x) {
    int count = 0, sum = 1;
    while (sum <= x) {
        count++;
        sum += 2 * count + 1;
    }
    return count;
}
```

Counting Votes with Formal Methods (p.26 of 40)

Extended Static Checker for Java

- ESC/Java developed at Compaq SRC (Leino et al), for finding common errors in Java programs:
 - null dereference errors
 - array bounds errors
 - type cast errors
- Uses translation to guarded command language & back-end automatic theorem prover (Simplify).
- It is neither sound nor complete—but very effective.
- Only partial compatibility with JML
- Development stopped in 2001??; released as open source (what license)??

Counting Votes with Formal Methods (p.27 of 40)

ESC/Java2

- New, open source version, developed by David Cok and Joe Kiniry
- Freely available for various platforms, see www.cs.kun.nl/sos/research
- Supports full JML (and Java 1.4), with wider support (model variables, pure methods, ...)
- Basis for various extensions & research projects
- Great for teaching & motivating formal methods! (range of tools is important)

Counting Votes with Formal Methods (p.28 of 40)

Testing with JML

- **Runtime assertion checking**
 - Complex test code is generated automatically from JML assertions
 - Good feedback on location & reason for violation (e.g. invariant violated in line ...)
- **Unit testing**
 - Only requires for every type a list of values/objects

KOA overview

-
- ```
graph LR; A[file I/O classes] --> B[core classes]; C[GUI classes] <--> B;
```
- File input: two XML files with:
    - Encrypted votes (candidate codes)
    - Voting lists (linking candidates to codes)
  - GUI input/output for voting official (fool proof)
  - Looks like a first year student assignment ...

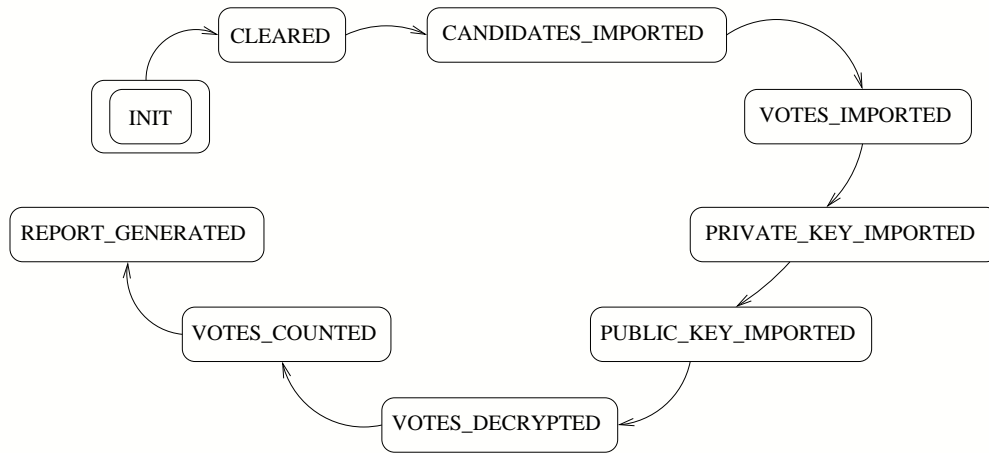
# KOA code statistics

|                         | File I/O | GUI   | core |
|-------------------------|----------|-------|------|
| number of classes       | 8        | 13    | 6    |
| number of methods       | 154      | 200   | 83   |
| number of lines of code | 837      | 1,599 | 395  |
| number of lines of spec | 446      | 172   | 529  |
| spec : code ratio       | 1:2      | 1:10  | 5:4  |

# KOA interface, for voting official



# KOA state diagram



Counting Votes with Formal Methods (p.33 of 40)

# State diagram in JML

```
/*@ spec_public int state;
@
@ invariant
@ state == INIT_STATE ||
@ state == CLEARED_STATE ||
@ state == CANDIDATES_IMPORTED_STATE ||
@ state == VOTES_IMPORTED_STATE ||
@ state == PRIVATE_KEY_IMPORTED_STATE ||
@ state == PUBLIC_KEY_IMPORTED_STATE ||
@ state == VOTES_DECRYPTED_STATE ||
@ state == VOTES_COUNTED_STATE ||
@ state == REPORT_GENERATED_STATE;
@
@ constraint
@ state == INIT_STATE ||
@ state == \old(state) ||
@ state == \old(state) + 1;
@*/
```

Counting Votes with Formal Methods (p.34 of 40)

## IV. Experiences

## JML Experiences

- First serious test of JML (with checking and testing) in simultaneous code & spec development
- JML works: many design & implementation errors detected early, but no 100% coverage
- JML forces to think about correctness early on: better to invest in assertions than in testcode
- Some detected deficiencies:
  - Not all (core) methods could be verified (due to Simplify issues)
  - Inconsistencies & ambiguities in API specs
  - Checking of forall/exists predicates insufficient
  - String semantics is incomplete; alias properties cumbersome

Counting Votes with Formal Methods (p.35 of 40)

Counting Votes with Formal Methods (p.36 of 40)

## Security Experiences

- No use of fancy crypto-based systems: infrastructure is lacking.
- Security is only partially technical, more procedural
- No transaction logs exist
- Multi-party oversight was effected: the ministry had:
  - increasingly strong “nothing to hide” attitude
  - healthy distance towards suppliers
  - rights to software
  - eventually released it as **open source under GPL** (see [www.ososs.nl](http://www.ososs.nl), like at [www.elections.act.gov.au](http://www.elections.act.gov.au))
- Early publication of hash of the source code became mess

Counting Votes with Formal Methods (p.37 of 40)

## V. Conclusions

## Organisational Experiences

- Strong presence of:
  - political context
  - social issues in technology adoption (useability, trust, etc.)
- Civil servants at the ministry are cautious:
  - Greatest sin: embarass their minister, e.g. via information leakage before parliament is informed
  - Despite contracts, building up trust is most important
- Much press attention (newspapers, radio, TV), generating further connections & contracts.
- Review of the whole project is underway, also for international discussions.

Counting Votes with Formal Methods (p.38 of 40)

## Conclusions

- Electronic voting can be good use of ICT; NL is early adopter
- Limited, cautious & low-tech experiment is wise
- Scaling to national level requires different set-up (esp. for authentication, and freedom/force/sale)
- Voting is (embarrassing?) challenge for CS-community: how reliable are our products?
- Vote counting software shows rapid application of research work in industrial setting
- Nice story of how academia can sometimes influence government and society in a postive way.
- Openness is essential, both for security & trust!

Counting Votes with Formal Methods (p.39 of 40)

Counting Votes with Formal Methods (p.40 of 40)