Democracy as a Critical System: Security, Formal Methods, and Elections

Joseph Kiniry
IT University of Copenhagen
applied formal methods

IT security
applied
formal
methods

IT
security
applied
formal
methods

mathematician

software
gineer

hacker

IT
security
Critical Systems

- biomedical
- automotive
- military
- avionics
- financial
- aeronautics
- nuclear
- transport
Democracy

voter registration

government legitimacy

voting systems

voting schemes

casting ballots

election outcomes

counting ballots

voter trust
Activism and Science
Voting Machines

- punchcard ballots
- physical locks
- lever machines
- dedicated primitive hardware
- mechanical ballot boxes
- off-the-shelf Windows machines

Thursday, 1 December, 2011
e-Voting Worldwide
e-Voting Worldwide
e-Voting in the EU
Computer-based Voting in The Netherlands

dedicated computer-based voting machines since the late 90s

people generally trust the government

experiments in remote voting for expats

hacking an election

recommendations to the government

tally system developed with formal methods

KOA

Thursday, 1 December, 2011
Computer-based Voting in Ireland

- PR-STV: novel social vote counting
- PowerVote: independent system testing
- CEV: last-minute secret purchase of €40M in Nedap machines
- Vótáil: scrapping e-voting at a cost of €55M

Thursday, 1 December, 2011
Computer-based Voting in Denmark

people generally trust the government

claim: no computers are used in voting

in truth: closed-source tally system used to compute final outcome

regular proposals to introduce e-voting

e-voting trials at the local level

DiVS

DemTech

Thursday, 1 December, 2011
Experiences in Hacking Voting Systems

- experiences with open source e-voting systems
- experiences with proprietary e-voting systems
- hacking remote elections
- hacking kiosk-based voting computers
- analyzing academic voting systems

Thursday, 1 December, 2011
Testing Voting Systems

most open source voting systems are not tested

most proprietary voting systems are not tested

“hard-core” testing is random testing of multiple implementations

random testing is no testing

how does one rigorously test a voting system?
Relating
The Law
to
Software
The State of e-Voting Software Today
The Law
char*M,A,Z,E=40,J[40],T[40];main(C){for(*J=A=scanf(M="%d",&C);
--
E;

printf("._");
for(;(A-=Z=!Z) || (printf("\n\n|",A-=39,C--);

Refinement Relation
In our tests, it counts correctly.

Overall Correctness Argument

Trust us, it works. How hard can it be, adding one over and over?
The State of Verified e-Voting Software Today
Table 1 gives an example - the numbers from the multi-member constituency of Esbjerg (Eastern Jutland).

1.2. Step Two: Determining Passing the Threshold

This step determines which parties are eligible for compensatory seats. This is done by checking if participating parties meet any of the three requirements. If the threshold for the party not to receive a compensatory seat is not met, the party is allowed to participate in the distribution of compensatory seats. If the number of votes corresponding at least to the provincial votes’ seat ratio is not met, the relevant numbers are shown in Table 2, which allows comparison of thresholds (2) and (3). The vote for the two parties in the single member constituency in the three electoral provinces is also comparable. In this case, the two parties below the 2 per cent hurdle are almost equally at the mercy of the threshold requirements, but lower parties below the 2 per cent hurdle are almost equally at the mercy of the threshold requirements.

Table 2: How the Parties that Failed to Qualify for Seats at Threshold (1) Failed on Threshold (2) and (3), November 13, 2007.

<table>
<thead>
<tr>
<th></th>
<th>All of Denmark</th>
<th>Copenhagen Metropolitan</th>
<th>Zealand Southern Jutland</th>
<th>Northern and Central Jutland</th>
</tr>
</thead>
<tbody>
<tr>
<td>Threshold (1)</td>
<td>n.a.</td>
<td>25,006</td>
<td>25,103</td>
<td>25,146</td>
</tr>
<tr>
<td>Threshold (2)</td>
<td>2 per cent</td>
<td>n.a.</td>
<td>n.a.</td>
<td>n.a.</td>
</tr>
<tr>
<td>Threshold (3)</td>
<td>2 per cent</td>
<td>58.199</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>The Parties</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>K. Christian People’s Party</td>
<td>30,013</td>
<td>5,613</td>
<td>7,636</td>
<td>16,686</td>
</tr>
<tr>
<td>K. New Alliance</td>
<td>97,290</td>
<td>40,243</td>
<td>30,356</td>
<td>25,696</td>
</tr>
</tbody>
</table>
A logical clock.

**Class chart LOGICAL_CLOCK**

- **Query**
  - What is the current time of this clock?
- **Command**
  - Advance the clock; update the clock's time.
- **Constraint**
  - The time must be non-negative.
  - Must support concurrent use by multiple clients.

**Informal EBON**

```plaintext
class_chart LOGICAL_CLOCK
explanation
"A logical clock."
query
"What is the current time of this clock?"
command
"Advance the clock; update the clock's time."
constraint
"The time must be non-negative."
"Must support concurrent use by multiple clients."
end
```

**Formal EBON**

```plaintext
deferred class LOGICAL_CLOCK
feature
my_time: INTEGER -- The current time of this clock.
-- What is the current time of this clock?
deferrad get_logical_time: INTEGER
-- concurrency: CONCURRENT
-- modifies: QUERY
ensure
Result = my_time;
end
deferrad advance -- Advance this clock's time.
-- concurrency: GUARDED
-- modifies: my_time
ensure
-- This clock's time has monotonically increased.
old my_time < my_time;
end
invariant
0 <= my_time;
end -- class LOGICAL_CLOCK
```

**JML**

```plaintext
public interface LogicalClock {
// The current time of this clock.
//@ public model instance _time;
//@ public invariant 0 <= _time;
/**
* @return What is the current time of this clock?
* @concurrency CONCURRENT
*/
//@ ensures _result == _time;
public /*@ pure @*/ long getLogicalTime();
/**
* Advance this clock's time.
* @concurrency GUARDED
*/
//@ assignable _time;
//@ ensures (* _time has been increased. *);
public void advance();
}
```

**Java**

```plaintext
public class LogicalClockImpl implements LogicalClock {
/** The current logical time. */
private long my_time = 0;//@ in _time;
//@ private represents _time <- my_time;

public long getLogicalTime() {
return my_time;
}
public void advance() {
my_time++;
}
}
```
Danish Law

Verified Software

Refinement Relation

Thursday, 1 December, 2011
If the input is as we characterized, then we guarantee a correct tally as output.

Overall Correctness

Argument

Proof is aggregate modular verification of system’s components.
Governments do not trust Verification
Governments think they trust Testing
Automated Testing that complements Formal Verification
Table 1 gives an example of the numbers from the multi-member constituency of Roskilde (Eastern Jutland).

1.2. Step Two: Determining the Passing Threshold
This step determines which parties are eligible for compensatory seats. This is done by checking if participating parties meet any of the three requirements. Thus, the Danish electoral system has not one but three different electoral thresholds and parties qualify for participation in the allocation of compensatory seats by any one of them. The three thresholds are:

1. winning a seat directly in any of the ten multi-member constituencies,
2. obtaining in two of the three electoral provinces a number of votes corresponding, at least, to the provincial vote seat ratio resulting in the calculation of these ratios the number of seats in the multi member constituencies in the electoral provinces, or questions, differing the provinces' compensatory seats, or
3. 2 per cent of the valid, national vote.

For parties that do not meet the first requirement (i.e. in 2007, it was two of nine participating parties), the relevant numbers are shown in Table 2, which allows a comparison of thresholds (2) and (3), and thus the votes for the two parties in question in the three electoral provinces as well as nationally.

Experience shows that threshold (3), the 2 per cent rule, is much more important than threshold (2), the vote seat ratio in two of three electoral provinces. Parties that meet the 2 per cent requirement will often also have met threshold (2) - whereas the case is quite the contrary - with parties below the 2 per cent hurdle almost inevitably will not meet any of the other requirements (as shown by the example of the Christian People's Party in 2007, which failed to cross any one of the three thresholds).

This experience illustrates how Danish political parties are not (any longer) primarily local or provincial in their support patterns.

1.2.3. Step Three: Allocating Compensatory Seats to Parties
This is the second step, since the electoral party list is grouped into five categories: proportional, the regional, the national, the local, and the extra-regional parties. The calculation (represented in Table 3) allocates the seats to parties which have qualified for participation in this allocation in strict proportionality to the number of votes obtained by these parties. The calculation is done on the basis of the so-called pure party quota, seats not allocated by the full

<table>
<thead>
<tr>
<th>Party</th>
<th>All of Denmark</th>
<th>Metropolitan</th>
<th>South</th>
<th>Northern and</th>
<th>Central Jutland</th>
</tr>
</thead>
<tbody>
<tr>
<td>Valid votes per multi-member constituency seat</td>
<td>n.a.</td>
<td>25,006</td>
<td>25,103</td>
<td>25,146</td>
<td></td>
</tr>
<tr>
<td>2 per cent of valid national votes</td>
<td>59,199</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

The Parties Votes:
- K. Christian People's Party: 30,013
- V. New Alliance: 97,295

The Law
Table 1 gives an example: the numbers from the multi-member constituency of Eastland.

### 1.2.2. Step Two: Determining of Passing the Threshold
This step determines which parties are eligible for compensatory seats. This is done by checking if participating parties meet any of three requirements. Thus, the Danish electoral system has not one, but three different electoral thresholds, and parties qualify for participation in the allocation of compensatory seats by any one of them. The three thresholds are:

1. winning a seat directly in any of the ten multi-member constituencies
2. obtaining in two of the three electoral provinces a number of votes corresponding at least to the provincial renewal ratio living in the calculation of these seats, the number of seats in the multi-member constituencies in the electoral province in question, subtracting the provincial compensatory seat, or
3. 2 per cent of the valid, national vote.

For parties that do not meet the first requirement (in 2007 it was two of ten multi-member seats).

<table>
<thead>
<tr>
<th>Name</th>
<th>Seats</th>
<th>Votes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Countrywide Party</td>
<td>2</td>
<td>30,013</td>
</tr>
<tr>
<td>Christian League</td>
<td>2</td>
<td>5,013</td>
</tr>
<tr>
<td>Social Alliance</td>
<td>2</td>
<td>7,635</td>
</tr>
<tr>
<td>Local Election</td>
<td>2</td>
<td>16,865</td>
</tr>
</tbody>
</table>

Table 2, How the Parties that Failed to Qualify for Seats at Threshold (1) Based on Threshold (2) and (3), November 1, 2011:

<table>
<thead>
<tr>
<th>Threshold 1</th>
<th>Seats</th>
</tr>
</thead>
<tbody>
<tr>
<td>21%</td>
<td>2</td>
</tr>
<tr>
<td>20%</td>
<td>2</td>
</tr>
<tr>
<td>19%</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Threshold 2</th>
<th>Seats</th>
</tr>
</thead>
<tbody>
<tr>
<td>15%</td>
<td>3</td>
</tr>
<tr>
<td>14%</td>
<td>3</td>
</tr>
<tr>
<td>13%</td>
<td>3</td>
</tr>
</tbody>
</table>

**Plural Knowledge**

- An individual person standing for election
- set Balloc, -- First preference ballots assigned to this candidate
- transfers: set Balloc, -- Second and subsequent preferences received
- surplus: set Balloc, -- Voters transferred to another candidate election
- wasted: set Balloc, -- Ballots non-transferable due to exhaustion of preferences
- outcome: Event, -- Election result for candidate and associated ballots

```java
// Non-transferable ballots
0 < #wasted implies outcome = WinnerNonTransferable or outcome = QuotaWinnerNonTransferable or outcome = EarlyLoserNonTransferable or outcome = SoreLoserNonTransferable
implies wasted in surplus
outcome = EarlyLoserNonTransferable or outcome = SoreLoserNonTransferable
implies wasted in votes + transfers
// Division of ballots into first preferences and transfers
no b: Balloc | b in votes + transfers
// Division of ballots into prizes for each candidate
all b: Balloc | b in votes + transfers implies this in b.assignees
// Selection of surplus ballots for re-distribution
surplus in votes + transfers
ElectroMethod = Plurality implies #surplus = 0 and #transfers = 0
0 < #transfers implies ElectroMethod = STV
// Calculation of surplus for PR-STV election
(outcome = Winner and ElectroMethod = STV) or (outcome = SurplusWinner or outcome = WinnerNonTransferable) implies
Scenarioquota + #surplus = #votes
outcome = Winner or outcome = SurplusWinner or outcome = WinnerNonTransferable
implies #transfers = 0
outcome = QuotaWinnerNonTransferable implies surplus in transfers
outcome = QuotaWinner or outcome = AbovetoQuotaWinner or outcome = QuotaWinnerNonTransferable
implies Scenarioquota + #surplus = #votes + #transfers
0 < #surplus implies outcome = SurplusWinner or outcome = AboveQuotaWinner or outcome = WinnerNonTransferable or outcome = WinnerNonTransferable implies Scenarioquota + #surplus = #votes + #transfers
outcome = EarlyLoser or outcome = TiedEarlyLoser or outcome = QuotaWinnerNonTransferable
implies EarlyLoserNonTransferable
if (this in Scenario Eliminated and not (votes + transfers < Scenario.threshold))
// All non-sort losers are at or above the threshold
outcome = TiedLoser implies Scenario.threshold <= #votes + #transfers
```
Table 1 gives an example—the numbers from the multi-member constituency of Islington North (UK)

1.2.2. Step Two: Determining of Passing the Threshold

This step determines which parties are eligible for compensatory seats. This is done by checking if participating parties meet any of these requirements. Thus far, the quota has been reached for those parties that have met the first threshold. The next two thresholds are then checked for eligibility.

2. parties that do not meet the first requirement in 2007 (they were two of nine participatig parties).

2.3. Step Three: Allocating Compensatory Seats to Parties

This is the stage at which the method is shared. The proportional representation allocation is based on the number of votes received in the respective constituency. Each party eligible for compensatory seats is then allocated a seat proportionally to the number of votes obtained.

All of Denmark Metropolitan Copenhagen Zealand Southern Jutland Northern and Central Jutland

<table>
<thead>
<tr>
<th>Threshold</th>
<th>Valid votes per multi-member constituency seat</th>
</tr>
</thead>
<tbody>
<tr>
<td>Threshold 1</td>
<td>26,906, 25,133, 25,146</td>
</tr>
<tr>
<td>Threshold 2</td>
<td>30,213, 5,613, 7,835, 16,865</td>
</tr>
<tr>
<td>Threshold 3</td>
<td>97,285, 40,241, 30,353, 20,096</td>
</tr>
</tbody>
</table>

---

**e-Voting Test Harness**

---

//-- An individual person standing for election

// If Candidate()

voted: Balle, -- First preference ballot assigned to this candidate

transfers: Balle, -- Second and subsequent preferences received

wasted: Balle, -- Ballots transferred to another candidate election

outcome: Event, -- Election result for candidate and associated ballots

// Non-transferable ballots

0 < @wasted implies outcome = Winner|NonTransferable or outcome = QuotaWinner|NonTransferable or outcome = EarlyLoser|NonTransferable or outcome = SoreLoser|NonTransferable

imlines wasted in surplus

outcome = Winner|NonTransferable or outcome = SoreLoser|NonTransferable

imlines wasted in votes + transfers

// Division of ballots into first preferences and transfers

no b: Ballot | b in votes & transfers

// Division of ballots into seats for each candidate

all b: Ballot | b in votes + transfers implies this in b, a, assignee

// Selection of surplus ballots for re-distribution

surplus in votes + transfers

Electromethod = Plurality implies surplus = 0 and transfers = 0

0 < @transfers implies Electromethod = STV

// Calculation of surplus for PR-STV election

(outcome = Winner and Electromethod = STV) or (outcome = SurplusWinner or outcome = Winner|NonTransferable) implies Scenarioquota + surplus = #votes

outcome = Winner or outcome = SurplusWinner or outcome = Winner|NonTransferable implies #transfers = 0

outcome = QuotaWinner|NonTransferable implies #transfers = 0

outcome = QuotaWinner|NonTransferable and outcome = AboveQuotaWinner or outcome = QuotaWinner|NonTransferable implies surplus in transfers

outcome = QuotaWinner or outcome = AboveQuotaWinner or outcome = QuotaWinner|NonTransferable implies Scenarioquota + surplus = #votes + #transfers

0 < surplus implies outcome = SurplusWinner or outcome = AboveQuotaWinner or outcome = Winner|NonTransferable or outcome = Winner|NonTransferable or outcome = QuotaWinner|NonTransferable or outcome = EarlyLoser|NonTransferable or outcome = EarlyLoser|NonTransferable if

SoreLoser|NonTransferable

// All non-losing persons are at or above the threshold

outcome = TiedLoser Implies ScenarioThreshold <= #votes + #transfers
Table 1 gives - as an example - the numbers from the multi-member constituency of Eastland (Eastern Jylland).

1.2.2. Step Two: Determining of Passing the Threshold

This step determines which parties are eligible for compensatory seats. This is done by checking if the participating parties meet any of these requirements. Thus, the Danish electoral system has not one, but three different electoral thresholds, and parties qualify for participation in the allocation of compensatory seats by any one of them. The three thresholds are:

1. winning a seat directly in any of the ten multi-member constituencies;
2. obtaining in two of the three electoral provinces a number of votes corresponding at least to the provincial vote-seat ratio existing in the calculation of these ratios the number of seats in the multi-member constituencies in the electoral provinces in question, excluding the provinces’ compensatory seats; or
3. 2 per cent of the valid, national vote.

For parties that do not meet the first requirement (in 2007 it was two of nine participating parties).

1.2.3. Step Three: Allocating Compensatory Seats to Parties

This is the decisive step, since it is here that the proportion, overall, national or upper tier allocation of all 1¼% seats takes place. The calculation (reproduced in Table 3 below) allocates the seats available to parties which have qualified for participation in this allocation instead proportionally to the number of votes obtained by these parties. The calculation is done on the basis of a so-called zero vote quota, or seats not allocated by the full

Table 2. How the Parties that Failed to Qualify for Seats at Threshold 1 (Based on Threshold 2) and (J) November 1 3, 2007.

<table>
<thead>
<tr>
<th>All of Denmark</th>
<th>Metropolitan Copenhagen</th>
<th>Southland Southern Jylland</th>
<th>Northern and Central Jylland</th>
</tr>
</thead>
<tbody>
<tr>
<td>Threshold 2: Valid votes per multi-member constituency seat</td>
<td>n.a.</td>
<td>26,906</td>
<td>25,133</td>
</tr>
<tr>
<td>Threshold 2: 2 per cent of valid national votes</td>
<td>n.a.</td>
<td>30,013</td>
<td>5,631</td>
</tr>
</tbody>
</table>

e-Voting Test Harness

Thursday, 1 December, 2011
Danish Law

Formally-generated Test Harness

Refinement Relation

Informal EBON

Formal EBON

JML

Java

In this example, the concepts identified through domain analysis are alarm, alarm clock, and logical clock. Their relationships are summarized in the EBON static diagram CONCEPTS AND RELATIONS in Listing 1. Their definitions are elided in this example. Each concept is summarized with an informal diagram. An informal diagram describes the concept and its interfaces in terms of queries, commands, and constraints. Queries and commands are collectively known as features. For example, the logical clock must store a time value and, in EBON terminology, support a query to determine the current time stored in the clock. A command is also necessary to monotonically advance the time stored in the clock. Furthermore, a constraint states that the time stored in the clock is always non-negative. Finally, the logical clock must also behave correctly while being used by multiple concurrent clients.

Listing 1:

Listing 2:

This interface and requirements are expressed using an EBON informal chart. Like most requirement languages, informal EBON uses structured English to denote analysis concepts and requirements. The EBON class chart shown in Listing 2 captures this information.
Unit Testing from Specs

90% coverage
Manual System Testing from Law

90% coverage with only a dozen system tests
System Testing from Law

for every unique election outcome
A Formal Model of Voting
A Parameterized Formal Model of Several Voting Schemes
An individual person standing for election.

\[
\text{Candidate}\{ \\
\text{votes: set Ballot, -- First preference ballots assigned to this candidate} \\
\text{transfers: set Ballot, -- Second and subsequent preferences received} \\
\text{surplus: set Ballot, -- Ballots transferred to another candidate; election} \\
\text{wasted: set Ballot, -- Ballots non-transferable due to exhaustion of preferences} \\
\text{outcome: Event -- Election result for candidate and associated ballots} \\
\} \\
\]

// Non-transferable ballots
0 < #wasted implies outcome = WinnerNonTransferable or
outcome = QuotaWinnerNonTransferable or
outcome = EarlyLoserNonTransferable or
outcome = SureLoserNonTransferable
(outcome = WinnerNonTransferable or outcome = QuotaWinnerNonTransferable)
implies wasted in surplus
(outcome = EarlyLoserNonTransferable or outcome = SureLoserNonTransferable)
implies wasted in votes = transfers
// Division of ballots into first preferences and transfers
no b: Ballot \b in votes & transfers
// Division of ballots into piles for each candidate
all b: Ballot \b in votes + transfers implies this in b assignees
// Selection of surplus ballots for re-distribution
surplus in votes + transfers

ElectroMethod = Plurality implies #surplus = 0 and #transfers = 0
0 < #transfers implies ElectroMethod = STV

// Calculation of surplus for PL-STV election
(outcome = Winner and ElectroMethod = STV) or
(outcome = SurplusWinner or outcome = WinnerNonTransferable)
implies
Scenario#quota = #surplus = Purposes
(outcome = Winner or outcome = SurplusWinner or
outcome = WinnerNonTransferable) implies #transfers = 0
(outcome = QuotaWinner or outcome = AboveQuotaWinner or
outcome = QuotaWinnerNonTransferable) implies surplus in transfers
(outcome = QuotaWinner or outcome = AboveQuotaWinner or
outcome = QuotaWinnerNonTransferable) implies
Scenario#quota + #surplus = #votes + #transfers
0 < #surplus implies (outcome = SurplusWinner or outcome = AboveQuotaWinner or
outcome = WinnerNonTransferable or outcome = QuotaWinnerNonTransferable)
(outcome = EarlyLoser or outcome = TiedEarlyLoser or
outcome = EarlyLoserNonTransferable) iff
(this in ScenarioEliminated and
not (votes + #transfers < Scenario#threshold))
// All non-lose losers are at or above the threshold
outcome = TiedLoser implies Scenario#threshold <= #votes + #transfers

Alloy Model
Law-Alloy Refinement
Rigorous System Test Generation
Core Concepts of Elections

scenario

candidate

ballot

event

method

election
Core Concepts

• candidate

• votes (set of ballots)

• transfers (set of ballots)

• surplus (set of ballots)

• outcome (event)

• ballot

• assignees (set of candidates)

• preferences (sequence of candidates)
Core Concepts

- scenario
- losers (set of candidates)
- winners (set of candidates)
- eliminated (set of candidates)
- threshold (integer minimum # of votes to not be a sore loser)
- quota (integer minimum # of votes for an STV or quota winner)
Core Concepts

• event, exactly one of...
  • Winner, QuotaWinner, CompromiseWinner, TiedWinner, TiedLoser, Loser, TiedEarlyLoser, EarlyLoser, TiedSoreLoser, SoreLoser

• election
  • candidates (set of candidates)
  • seats (integer)
  • method (plurality or STV)
  • ballots (integer # of unspoiled ballots)
Generating Scenarios

• goal: generate and characterize every possible non-isomorphism scenario

• election method, # candidates, # seats

• example outcomes

• WL or WL in two candidate plurality

• SSSLLLLLLLWWW with 10 candidates and 1 seat in STV

• scenarios as lemmas

• “I bet there can’t be an election outcome like this!”
Coupling Systems

- couple Alloy to jUnit
- generate and save system tests in generic format for reuse across implementations
- perform code coverage analysis
- characterize system correctness
- identify suspicious parts of an implementation
Ongoing Results

- generated all scenarios for up to 7 candidates in PR-STV using several months of CPU time
- 99.9% code coverage
- early results after only two days of CPU time detected two cases missed in scenario analysis
- zero bugs detected in verified counting system
Summary of Current Affairs

• formally specified, validated, and verified election tally software systems for US, NL, IE, and DK
• traceable refinement from law—interpreted as concepts, features, and requirements—to specifications, software, and proofs
• automatic verification using ESC/Java2
• automated unit tests with 97% coverage
• manual system tests with 97% coverage
• automated system tests with 100% coverage
• all research and development done in “spare time”
Next Steps

• formal model of elections
• system model that includes people, parties, bureaucrats, government
• trust-by-design
  • software engineering in the face of an adversarial customer (gov. and citizens)
• logic-based voting scheme
• couple LF s to implementation
Danish Council for Strategic Research
Programme Commission on
Strategic Growth Technologies

DemTech

5 years
17M direct
32M total

Basin (ETHZ)
Ryan (Lux)

Siemens
Aion Assembly

Schürmann
Kiniry
Markussen

Fredericksberg
Aarhus
Copenhagen

Thursday, 1 December, 2011
Thanks to Collaborators

- KOA
- Dermot Cochran, Fintan Fairmichael, Engelbert Hubbers, Alan Morkan, Martijn Oostdijk
- Vótáil
  - Dermot Cochran
- DiVS
  - Dermot Cochran, Ólavur Kjølbro
See DemTech.dk for more information