Analysis of a distributed e-voting system architecture against quality of service requirements

J Paul Gibson, Eric Lallet, Jean-Luc Raffy
Institute Telecom Sud Paris
Evry, France
Paul.Gibson@it-sudparis.eu

Abstract

In this paper we propose that formal modelling techniques are necessary in establishing the trustworthiness of e-voting systems and the software within. We illustrate how a distributed e-voting system architecture can be analysed against quality of service requirements, through simulation of formal models. A concrete example of a novel e-voting system prototype (for use in French elections) is used to justify the utility of our approach. The quality of service that we consider is the total time it takes for a voter to record their vote (including waiting time). The innovative aspects of the e-voting system that required further research were new requirements for voting anywhere and re-voting; and the potential for undesirable interactions between them.

1. Background and Motivation

1.1 The E-voting Problem

Computer technology has the potential to modernise the voting process and to improve upon existing systems; but it introduces concerns with respect to secrecy, accuracy, trust and security [15]. Despite ever-increasing uncertainty over the trustworthiness of these systems many countries have recently chosen to adopt e-voting, and this has led to their analysis from a number of different viewpoints: usability [17], trustworthiness [22], transparency [24] and risks [27].

Many of the problems in the domain of e-voting have arisen because of poorly specified requirements and standards documents, and the inability to carry out meaningful verification [23]. This is not a good reflection of best practice in the engineering of software. We propose that the use of formal methods [5], following a model-driven development process[29], is a good reflection of best practice in the development of critical software. In this paper we analyse the utility of such an approach in the critical step between requirements and design, where we analyse whether alternative architectures, for an e-voting system, are able to support a critical quality of service.

1.2 Formal Modelling

Formal methods are a tool for achieving *correct* software: that is, software that can be proven to fulfil its requirements. Building a formal model improves understanding. The modelling of nondeterminism, and its subsequent removal in formal steps, allows design and implementation decisions to be made when most suitable [8].

There are three important aspects to the use of formal methods for requirements and design modelling: Firtly, the method must be compositional so that incremental development is supported. A formal object oriented approach illustrates the advantages of an incremental modelling process when using simulation for validation and verification [10]. Secondly, the method must offer a means of specifying operational requirements for animation during validation, where it is important to be able to control execution of a subset of system behaviour whilst a simulator controls the other parts in a manner which corresponds to how the system behaves, or should behave, in the real world [12]. Finally, the method must offer a means of specifying logical requirements: both safety properties which state that *bad things can never happen* and liveness properties which state that *something good will eventually happen* [14].

Given the different roles played by the requirements and design models, we believe that there is a need for a number of different modelling languages when verifying designs against different types of requirements. Of course, this poses the problem of how to ensure that the different models are consistent and how to integrate them into a coherent whole [11]. This is beyond the scope of this paper, where we focus on the animation and simulation aspects of formal modelling and the verification of quality of service, where only a single modelling language is required.

1.3 Meeting New Requirements: Practical Demonstration

The work presented in this paper is part of an applied research project — "Sécurité et Audit du Vote Electronique" (SAVE) — funded by the French *Agence National*

978-0-7695-3372-8/08 $25.00 © 2008 IEEE
DOI 10.1109/ICSEA.2008.18

de la Recherche (ANR), in which the objective is to develop a prototype for an innovative e-voting system for use in France. In our chosen system, we have two clear voting innovations (for France) that the system will be required to support. Firstly, we wish to allow voters to be able to go to any polling station in order to vote; currently they are required to go to a particular station. We refer to this as a *VoteAnywhere* feature. Secondly, we wish to allow voters to be able to re-vote so that a previously recorded vote is overwritten by a new vote; currently voters have no way of changing their vote in such a way. We refer to this as a *ReVote* feature. We examine these new requirements in more detail in section 3. In particular, in section 3.4, we examine the quality of service that a voter expects from a voting system in order for them to be able to record a vote in a reasonable amount of time.

Our main research contribution is to demonstrate that it is possible to formalise the functional requirements (*VoteAnywhere* and *ReVote*) and to analyse, through simulation of formal models, the quality of service offered by different architectures for distributed e-voting systems (that meet these functional requirements).

2. Quality of service analysis: simulation with Estelle

For simulation and analysis of quality of service requirements we have chosen to use Estelle [7], a Formal Description Technique standardised by ISO [18]. Although this technique is not as popular as other better known formal methods, it is well suited to the analysis task outlined in this article. Its main application field is the formal specification of distributed systems using communication protocols, and it permits a clear split between the definition of the global architecture of the system and the internal behaviour of its components.

An Estelle specification describes a collection of hierachical communicating components. A component is an instance of a generic module definition composed of a single header definition and one or more associated body definitions. These instances may be statically or dynamically created.

The header definition describes the external communication part of the module and specifies a synchronous parallel or a non-deterministic serial execution. The communication interface of a module is defined by ports called interaction points, each of which refers to a channel which refines two sets of interactions (messages sent and received). Nested modules can also communicate by sharing exported variables.

The body definition describes the behaviour of the component. It uses the extended finite state machine (EFSM) paradigm. It is composed of three parts: a declaration part, an initialisation part and a transition part where the EFSM is described.

An Estelle development can be divided into four main phases: specification, static analysis, simulation and implementation(see figure 1). In this paper we focus on analysis resulting from the simulation phase.

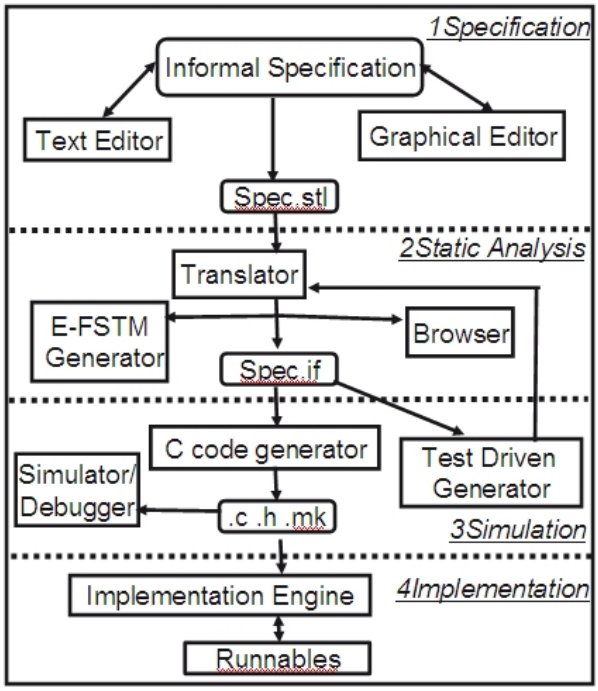

Figure 1. Phases of an Estelle Development

3. Our E-voting System Requirements

In this section we give an informal description of our innovative e-voting system requirements. Analysis of different possible architectures (modelled using Estelle) for meeting these requirements will be reviewed in section 4.

3.1 Vote Anywhere

Currently in France, each voter is assigned to a unique polling station where their name is found on an electoral list. In order to vote, their identity is authenticated and consequently they may be authorised to place a ballot in the urn (provided that they have not already done so). Checking that someone has not already voted involves a simple protocol: just after a voter is allowed to place a ballot in the urn they are required to sign next to their name on the voter list. If they have already signed then they are considered to have already registered their suffrage and will not be allowed to place the ballot in the urn. In the exceptional case

where a voter refuses to sign, an election official signs on their behalf and a public announcement of this is made.

Our new requirement is that a voter should not be obliged to go to a particular polling station in order to vote; they should be able to go to any authorised voting station. There are two main issues that need to be addressed when meeting this requirement. Firstly, how do we know that the voter has not already voted (at a different polling station)? Secondly, how do we generate the correct ballot (list of vote options) where this depends on the constituency in which the voter is registered to vote and independent of the station where they went to vote?

3.2 Re-vote

The notion of allowing a voter to change their mind and to re-cast a suffrage is one which often arises when considering remote electronic voting. One could argue that allowing re-votes could hinder vote coercion [20] in this situation.

This new required functionality would pose many problems for the traditional French voting system: after an elector drops a ballot into the urn then there is no way of accessing the contents of the urn until the voting process is closed. This mechanism is used to ensure that no-one can surreptitiously add or remove votes. We note that the urn is transparent (which assists verification of the fact that no votes are recorded before voting is open). Transparency means that, in order to meet the additional requirement of secrecy, French voters are required to place their ballots in envelopes before they place them in the urn.

The current system could support re-voting provided there was a means of finding the ballot of a particular voter in the urn without disclosing the suffrage recorded on any of the ballots. We refer to such a ballot as being signed. Then we have two obvious choices of protocol for meeting the requirement:

> **(1)** When an elector wishes to place a ballot in the urn, they are required to sign it. Then remove all previous ballots in the urn that share this signature (there should be only one). Finally, place the new ballot in the urn. We note that if voter anonymity is required then the signature must be done in such a way that one can find the vote of a particular voter without disclosing the choice that they have actually recorded.
>
> **(2)** Timestamp the ballots and when the recording of suffrages is terminated then remove all except the most recent ballot of every voter. The notion of timestamp is left as abstract, for the moment. It is to facilitate the ordering of ballots of a single elector based on the time they were deposited in the urn.

3.3 Re-vote Anywhere

We must consider the interdependency between the *VoteAnywhere* and *Revote* requirements, as both features must be offered in our chosen system. The main issues are whether being able to vote multiple times at different polling stations is functionally possible and, if so, can we continue to offer an acceptable quality of service to the voters?

The interdependency, in this case, is not bad in the classic sense of a requirements feature interaction[9]: analysis will show that the *Revote* feature can help in the process of designing a solution to the *VoteAnywhere* non-functional requirement that a denial of service attack on the network cannot compromise the voting system in meeting a minimum quality of service.

3.4 Quality of Service Requirements

The media has recently reported quality of service issues with many voting systems (both traditional and electronic) around the world. In France, for example, a significant number of voters were required to queue for more than an hour in order to have access to newly introduced electronic voting machines; and as a consequence some voters left without recording a vote. The main problem was due to a number of traditional voting booths being replaced by a single voting machine. However, it appears that no-one thought to analyse whether the new electronic system offered the same quality of service as the traditional system.

The most significant threat to being able to meet quality of service requirements is a complete denial of service, where users of the system are unable to execute core functionality. In the case of e-voting machines, such a denial of service would prohibit anyone from recording a vote. When an e-voting system relies on components that are accessed across a network then a significant denial of service threat would arise if the network was not reliable (or if it was not resistant to attacks). This is a well documented concern for remote voting, including internet voting [19]. However, it is also a concern in our chosen system where the network is a key component.

The main advantage of our approach — reported in this paper — is the ability to reason about e-voting quality of service (including denial of service) early in the development process. Analysis of such issues should be done as soon as possible — which means verifying that any proposed high-level design (architecture) is able to meet the quality of service requirements. Making a choice between alternative architectures should not be done without having first completed a verification of such requirements. This is the work that is reported in section 4.

4 Formal Modelling of Alternative Architectural Solutions

4.1 Quality of Service Analysis Using Estelle

In order to reason about quality of service properties in Estelle we are obliged to simulate behaviour of our proposed architectures in different environments. The standard approach is to identify key environmental parameters and to analyse the system's performance as these parameters change. In our case, we have two key dynamic parameters: the distribution over time of voters attempting to vote and the distribution over time of network inoperability.

For voter distributions we had access to a large bank of data from which we were able to generate a typical distribution curve (see the top graph in figure 2). For network operation, we had no data from which we could generate typical distribution curves of network downtime for voting systems. Consequently, we were obliged to use more generic information from our network providers about the types of distribution curves that their networks would offer under normal conditions (not associated with e-voting). Then, in order to stress test the system, we chose to align peak voting times with the peak network downtimes.

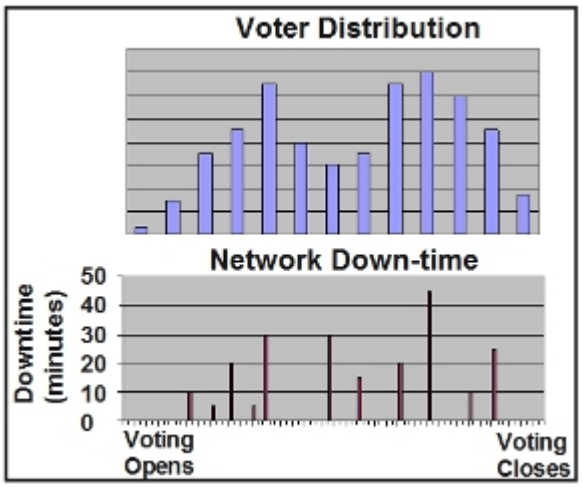

Figure 2. Key Simulation Distributions

The key role of the simulation is to show that certain architectures will not be able to meet the quality of service requirements. The architectures that are verified to meet the requirements (through simulation) can progress for further development, but need to be more thoroughly analysed during later development steps.

In the simulation, we modelled other important static parameters whose values, mostly based on observation of real polling stations, remain constant during execution:

- **Time for the elector to identify themselves**: chosen randomly from a normal distribution of between 10 and 20 seconds.
- **Time for the elector to select their option(s)**: chosen randomly from a normal distribution of between 20 seconds and 2 minutes.
- **Time for the elector to enter an empty voting booth**: chosen randomly from a normal distribution of between 10 seconds and 30 seconds.
- **Transmission delay on network**: 0 seconds, as we consider that this delay time is insignificant when the network is operational.
- **Number of polling stations**: 2, in order to simulate electors going to vote at a station which is not in the constituency where their vote will be counted.
- **Number of polling booths per station**: 3, which corresponds to a typical situation.
- **Number of electors per polling station**: 1000, which corresponds to the French legal requirement.
- **Percentage of abstentions**: 33, which corresponds to a typical French election.
- **Percentage of non-local votes (***VoteAnyWhere***)**: 20, chosen intuitively to be a realistic worse case scenario.
- **Percentage of** *Revotes*: 5, chosen intuitively to be a realistic scenario.

The simulation had to model the location of the key data within the architecture, so that inoperation of the network would result in communication delays and have a possibly negative impact on the quality of service. All simulations meet the *VoteAnywhere* requirement and they all used the same key parameter distributions and values (except the fourth in which the network is considered to be totally reliable and so the inoperation time is set to the constant value 0.)

The first simulation (see figure 3) is used to demonstrate that voting anywhere can lead to quality of service problems if the re-vote option is not offered. We give further details in section 4.2.

The second and third simulations (see figures 4 and 5, respectively) combine voting anywhere with revoting: the difference between them being that in the second case the candidate lists for all other polling stations are stored centrally (across the network) whilst in the third case the candidate lists for all polling stations are stored as local copies. We give further details in sections 4.3 and 4.4.

The fourth and final simulation (not illustrated) is used as a baseline for performance analysis — the network is

modelled as being completely reliable. It is unreasonable to expect the network component to be perfect but this simulation demonstrates that we can approach the performance of such an ideal system. We give further details in section 4.5.

4.2 Simulation 1: Unreliable Network with no *ReVote*, and networked electoral lists and candidate lists

The key aspect of this architecture (in figure 3) is that re-voting is not permitted and consequently we introduce a centralised voter list, accessed across the network, which records whether an elector has already voted in order to stop them from voting multiple times. The local urn is used to store the votes before they are transmitted across the network for counting at a global urn. The local electoral list (and vote options) are optimisations which allow authentifcation of voters who go to their local polling station (and generation of their ballot) to be done without having to access information across the network.

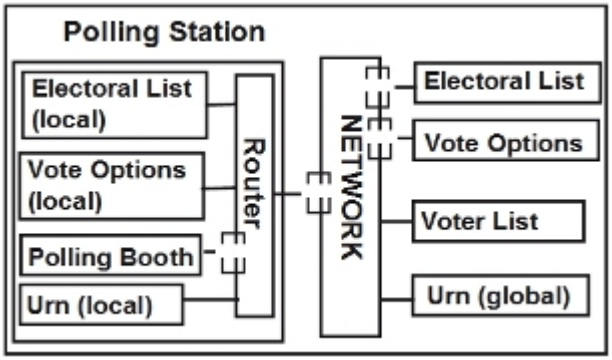

Figure 3. Simulation 1

The formal Estelle specification is available on request. We note the following properties that can be deduced from the diagram: each polling station is composed from five components, there are multiple polling booths in each polling station (connected to a single router), there are multiple polling stations connected to a single network, there are multiple electoral lists connected to a single network, and there are multiple vote options connected to a single network.

4.3 Simulation 2: Unreliable Network with *Revote*, and local and networked candidate lists

The key aspect of this architecture is that there is no need for a networked voter list because re-voting is permitted. As electors can vote multiple times, we introduce a timestamp in each polling booth so that only the most recent votes are counted when the votes are transferred to the global urn.

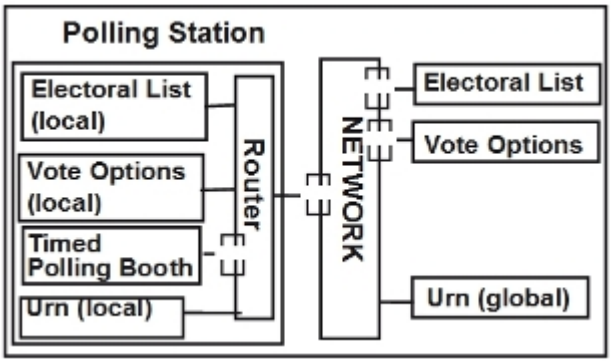

Figure 4. Simulation2

4.4 Simulation 3: Unreliable Network with *Revote*, no voter list and all candidate lists local

The key aspect of this architecture is that the candidate lists for all constituencies are stored locally and are not networked. Thus, only the global urn is on the network and so no voter is dependent on the network in order to vote. The network is now used only to transfer votes from the polling station to the global urn in order to be counted (at the appropriate constituency).

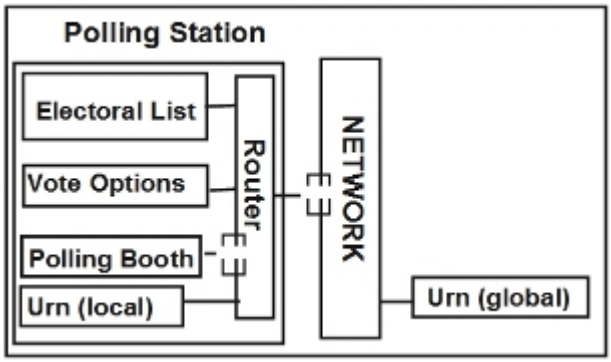

Figure 5. Simulation3

4.5 Reliable Network: revote where all architectures have equivalent performance

As we have chosen to model no transmission delays in a reliable network, there is no difference between access to a

database that is stored locally at a polling booth, or stored at a central location remote from a polling booth. Thus, we get the same quality of service results for each of our three architectural choices when the network is never inoperable. This simulation was executed by flattening the network downtime distribution curve to zero at all times.

5. Quality of Service Simulation Analysis

In figure 6 we see the graph of the voter waiting time distributions for each of our four simulations.

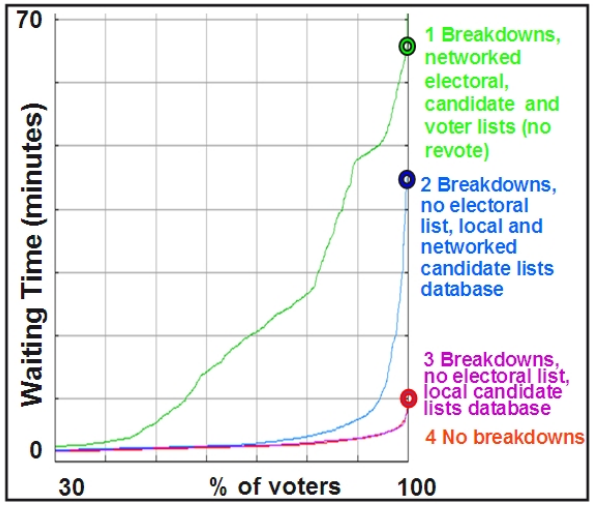

Figure 6. Simulation Waiting Time Graph

In simulation 4, with no network breakdowns, we see that all voters get to vote within a reasonable time (less than 10 minutes). This is considered to be an acceptable quality of service. In simulation 1, where revoting is not permitted, we see that less than half the voters get to vote within this reasonable time. This is not considered an acceptable quality of service. In simulation 2, where revoting is allowed we see that a small but significant number of voters have to wait much longer than 10 minutes. This is not acceptable because there is a large risk that a number of these voters will leave the voting station before recording a vote. In simulation 3, where the network is used only in the transfer of votes, we approach the quality of service of simulation 1.

In the table in figure 7 we summarise the key statistics for each of the simulations involving an unreliable network. We conclude that architectures 1 and 2 should be rejected as inappropriate for further development.

Waiting Time	Simulation 1	Simulation 2	Simulation 3
Minimum	23s	44s	45s
Maximum	71m45s	44m50s	8m11s
Mean	16m16s	3m3s	2m19s
Median	9m26s	2m15s	2m9s

Figure 7. Simulation Waiting Times

6. Related Work

In *Verification and Validation Issues in Electronic Voting* [4], definitions for verifiability and validity, in the context of e-voting, are proposed. Most research on e-voting and verification using formal modelling has focussed on particular voting protocols and the means of individual voters verifying that a count was carried out without any malicious manipulation of the votes[21].

Research on verification of the software in e-voting systems has, until quite recently, focussed on security protocols. For example, in the paper by Groth [16] we see that voting protocols are considered secure if they satisfy requirements such as privacy, accuracy, robustness, etc. Delaune et al. [6] formalise a subset of e-voting requirements and verify whether the particular e-voting protocols meet the requirements. They do not address quality of service.

There has been little published research on the use of formal methods for verification of other components in the voting system. Poppleton presents a formal specification (in the Z language) for a simplified version of an algorithm for counting using STV[28]. Research by Carew et al. [3] presents an empirical study which compares the comprehensibility of two e-voting requirements specifications: a formal specification and an informal specification. Cansell et al. [1, 2] recommend application of formal methods for guaranteeing tamper-evident storage of votes and secure voting machine interface development.

Mercuri [25] examines the problems of verifying against codes of laws. However, the use of formal methods for verification is not discussed in any detail. More recently, the problems associated with the verification of COTS software components in e-voting machines is discussed by Mercuri [26]. However, emphasis is placed on testing for verification rather than on the possibility of using formal methods.

7. Conclusions and Future Work

A major problem with e-voting systems is that they need to be both trustworthy and trusted [13]. At each step in the development process, there is a risk that bugs will be introduced into the system. The move from requirements specification to a high-level design is difficult to get right and formal verification techniques should be used wherever possible. Without formal techniques the systems being developed are less trustworthy and open to criticism.

We have shown that quality of service requirements can be analysed through simulation of high-level architectural

models. This approach would certainly complement the formal verification of logical properties using model checkers and/or theorem provers.

Through our simulations, we have helped guide design decisions made by the e-voting system developers. By demonstrating, early in the design process, the inappropriatness of certain architectures (with respect to their inability to meet quality of service requirements when the underlying network is not perfect) we have significantly aided the design process. Furthermore, by providing formal models we are more confident that the final system will meet the innovative functional requirements of *VoteAnywhere* and *Revote*.

References

[1] D. Cansell, J. P. Gibson, and D. Méry. Formal verification of tamper-evident storage for e-voting. In *SEFM*, pages 329–338. IEEE Computer Society, 2007.

[2] D. Cansell, J. P. Gibson, and D. Méry. Refinement: A constructive approach to formal software design for a secure e-voting interface. *Electr. Notes Theor. Comput. Sci.*, 183:39–55, 2007.

[3] D. Carew, C. Exton, J. Buckley, M. McGaley, and J. P. Gibson. Preliminary study to empirically investigate the comprehensibility of requirements specifications. In *Psychology of Programming Interest Group 17th annual workshop (PPIG 2005)*, pages 182–202, University of Sussex, Brighton, UK, 2005.

[4] O. Cetinkaya and D. Cetinkaya. Verification and validation issues in electronic voting. *The Electronic Journal of e-Government*, 5(2):117–126, 2007.

[5] E. M. Clarke and J. M. Wing. Formal methods: state of the art and future directions. *ACM Comput. Surv.*, 28(4):626–643, 1996.

[6] S. Delaune, S. Kremer, and M. Ryan. Coercion-resistance and receipt-freeness in electronic voting. In *CSFW06: Proceedings of the 19th IEEE workshop on Computer Security Foundations*, pages 28–42. IEEE Computer Society, 2006.

[7] M. Diaz, J. P. Ansart, and J. P. Couriat, editors. *Formal Description Technique Estelle: Results of the Esprit Sedos Project*. Elsevier Science Inc., New York, NY, USA, 1989.

[8] J. P. Gibson. *Formal Object Oriented Development of Software Systems Using LOTOS*. Thesis CSM-114, Stirling University, Aug. 1993.

[9] J. P. Gibson. Feature requirements models: Understanding interactions. In B. Dini and Logrippo, editors, *Feature Interactions in Telecommunication Networks IV*, pages 46–60. IOS Press, 1997.

[10] J. P. Gibson. Formal object oriented requirements: simulation, validation and verification. In H. Szczerbicka, editor, *Modelling and Simulation: A tool for the next millenium , vol II, ESM99*, pages 103–111. SCS, 1999.

[11] J. P. Gibson and D. Méry. A unifying model for specification and design. In F. Galmiche, Bashoun and Yonezawa, editors, *Proceedings of the Workshop on Proof Theory of Concurrent Object Oriented Programming*. Linz (Austria), July 1996.

[12] J. P. Gibson, Y. Mokhtari, and D. Méry. Animating formal specifications — a telephone simulation case study. In H. Szczerbicka, editor, *Modelling and Simulation: A tool for the next millenium , vol II, ESM99*, pages 139–146. SCS, 1999.

[13] Gibson, J. P. E-voting and the need for rigorous software engineering — the past, present and future. In J. Julliand and O. Kouchnarenko, editors, *B*, volume 4355 of *Lecture Notes in Computer Science*, page 1. Springer, 2007.

[14] Gibson, J. P. and D. Méry. *Fair objects*, pages 122–140. Horwood Publishing, Ltd., Chichester, USA, 1999.

[15] D. Gritzalis, editor. *Secure Electronic Voting*, volume 7 of *Advances in Information Security*. Kluwer Academic, 2003.

[16] J. Groth. Evaluating security of voting schemes in the universal composability framework. In M. Jakobsson, M. Yung, and J. Zhou, editors, *Applied Cryptography and Network Security*, volume 3089 of *Lecture Notes in Computer Science*, pages 46–60. Springer, 2004.

[17] P. S. Herrnson, B. B. Bederson, B. Lee, P. L. Francia, R. M. Sherman, F. G. Conrad, M. Traugott, and R. G. Niemi. Early appraisals of electronic voting. *Soc. Sci. Comput. Rev.*, 23(3):274–292, 2005.

[18] ISO/IEC. Estelle: A formal description technique based on an extended state transition model. Technical Report ISO 9074, Information technology - Open Systems Interconnection, 1997.

[19] D. Jefferson, A. D. Rubin, B. Simons, and D. Wagner. Analyzing internet voting security. *Commun. ACM*, 47(10):59–64, 2004.

[20] A. Juels, D. Catalano, and M. Jakobsson. Coercion-resistant electronic elections. In *WPES '05: Proceedings of the 2005 ACM workshop on Privacy in the electronic society*, pages 61–70, New York, NY, USA, 2005. ACM.

[21] C. Karlof, N. Sastry, and D. Wagner. Cryptographic voting protocols: a systems perspective. In *SSYM'05: Proceedings of the 14th conference on USENIX Security Symposium*, pages 3–3, Berkeley, CA, USA, 2005. USENIX Association.

[22] M. McGaley and Gibson, J. P. E-Voting: A Safety Critical System. Technical Report NUIM-CS-TR-2003-02, NUI Maynooth, Computer Science Department, 2003.

[23] M. McGaley and Gibson, J. P. Verification and maintenance of e-voting systems and standards. In *8th European Conference on e-Government*, 2008. to appear.

[24] M. McGaley and J. McCarthy. Transparency and e-Voting: Democratic vs. Commercial Interests. In *Electronic Voting in Europe - Technology, Law, Politics and Society*, pages 153 – 163. European Science Foundation, July 2004.

[25] R. T. Mercuri and L. J. Camp. The code of elections. *Communications of the ACM*, 47(10):52–57, 2004.

[26] R. T. Mercuri, V. J. Lipsio, and B. Feehan. COTS and other electronic voting backdoors. *Commun. ACM*, 49(11):112, 2006.

[27] P. G. Neumann. Inside risks: risks in computerized elections. *Commun. ACM*, 33(11):170, 1990.

[28] M. Poppleton. The single transferable voting system: Functional decomposition in formal specification. In H. McGloughlin and G. O'Regan, editors, *IWFM*, Workshops in Computing. BCS, 1997.

[29] B. Selic. The pragmatics of model-driven development. *IEEE Softw.*, 20(5):19–25, 2003.