# Specification of the Control Logic of an eVoting System in UML: the ProVotE experience

Roberto Tiella, Adolfo Villafiorita, and Silvia Tomasi

Automated Reasoning Systems Division
Center for Scientific and Technological Research (ITC-irst)
{tiella,adolfo,sitomasi}@irst.itc.it
WWW home page: http://sra.itc.it/provote

**Abstract.** The ProVotE project aims at actuating art. 84 of law 2 - 5/3/2003 of the Autonomous Province of Trento (Italy), which promotes the introduction of e-voting systems for the next provincial elections.
The goals of the project are building an e-voting system and defining the related process and certification issues in order to deliver a solution which is compliant with the italian electoral law, that is trusted and usable by the voters, and that implements the security and dependability requirements of the best international practices and experiences.
This paper presents some of the issues and challenges we faced during the development of the e-voting prototype which was tested by about 8500 voters in three different trials held during the elections for the local representatives. In particular we show how we integrated UML in the development process of the e-voting machine. We also highlight how we extended existing tools to support the formal verification of the UML specifications and how we automated the generation of a critical component of the e-voting machine using UML statecharts.

## 1 Introduction

The use of new technologies to support elections has been and is the subject of great debate. Not surprisingly, there are both advocates of the benefits it can bring - such as improved speed and accuracy in counting and accessibility - and people more concerned with the risks it poses, such as digital divide, violations to secrecy and anonymity, alteration of the votes and of the results (either because of malicious attacks or simply because of bad design and coding). See, for instance, [1–5] for a more in depth view on some of the topics mentioned above.

For historical and political reasons the Province of Trento (also PaT, from now on) benefits of special autonomy and it determines by its own legislation how the council and the President of the province are elected. Systems for transmitting results electronically from the polling stations to the Electoral Service have been in use in the Province for several years now. (In Italy PaT is among the earliest adopters.) The Province has now decided to move a step further and art 84 of PaT law 2/2003 promotes the introduction of forms of e-voting for

the next provincial elections, to be held at the end of 2008. To actuate the law PaT is sponsoring the ProVotE project, which has the goal of ensuring a smooth transition to the new way of voting.

This paper reports on some of the technological activities carried out during the development of the ProVotE e-voting prototype, which has been significantly based on UML. Matter of fact, UML has been used to model the electoral processes. Use cases and statecharts have been used during the specification phase to agree with the Electoral Service on the requirements of the e-voting machine. The source code of the core logic of the machine is, in fact, automatically generated from UML statecharts refining those inspected and approved by the Electoral Service. The UML statecharts have also been automatically translated into the NuSMV input language to formally verify them [6]. To effectively support such approach we adapted and developed tools to support the automatic translations.

## 2 Voting Procedures in Italy and e-voting Experimentations

Simplifying both on the law and on the procedures for the sake of presentation, voting in Italy happens as follows:

1. **identification and registration of the voter.** At the polling station the voter is usually required to show his/her ID card and the electoral card. If the name of the voter is present in the electoral list of the polling station, the voter is registered, the electoral card stamped, and the voter is admitted to voting.
2. **casting a vote.** The voter is given a ballot and a pencil and is shown a cabin where the vote can be cast in secrecy. Secrecy is both a right and a duty. The Italian law and procedures are aimed at ensuring that the voter cannot make his/her vote manifest to other people.

At the end of the voting day, the ballot boxes are opened and the counting procedure starts:

3. **counting.** Votes are counted and the results tabulated in special registers.
4. **transmission of the results.** When all the ballots have been tabulated, the results are transcribed in various paper documents and transmitted to the offices responsible of aggregating all the data.
5. **sum and proclamation of the elected representatives.** All the data coming from the different polling stations are counted and seats assigned according to algorithm defined by the law. Data are then made available to the general public.

Various experimentations have been conducted in Italy to introduce new technologies in the precincts. The largest trial, so far, was sponsored by the central government, and concerned a system for automating steps 3 and 4 above.

The system, operated by specially appointed technicians, was installed in 47 precincts at the last European elections and repeated at the last political elections (2006). Little, however, is known about the results of the experimentation. See [7] for some more details.

*Proper* e-voting experimentations (i.e. including step 2) have been conducted at the local level, usually on a small scale, in experimentations which seem to have had little continuity and/or on which information is scarce. We mention San Benedetto del Tronto (2000), trials sites in Avellino (2001), Campobasso (2001), Cremona (2002, 2006), Ladispoli (2004), Specchia (2005) [8, 9]. Other experimentations have been conducted in Valle D'Aosta, Friuli Venezia Giulia, and Milan.

Independently from their scope, past experiences demonstrate the importance of providing usable, reliable, safe, and secure systems also for trials, as trust on the systems is built also during experimentations. Proper development of prototypes, therefore, not only plays a key role in contributing to the quality of the final product, but it is also an essential step for project success.

## 3 The ProVotE Project and Motivations

ProVotE has the goal of ensuring a smooth transition to e-voting in Trentino, eliminating risks of digital divide and providing technological solutions which support, with legal value, steps 1 to 5 above, namely from voting to publication of the results.

The project includes partners from the public administration (Provincia autonoma di Trento, Regione Trentino/Alto-Adige, Consorzio dei Comuni Trentini, Comune di Trento, IPRASE), research centers and academia (ITC-irst, Faculty of Sociology of the University of Trento, Fondazione Graphitech), and local industries (Informatica Trentina) and is co-lead by the Electoral Service of the Autonomous Province of Trento and by ITC-irst. Project leadership by the Public Sector, in our opinion, among other advantages, helps tackling the issue of potential conflicts of interests by private industries, see e.g. [10].

The project is organized in three lines of activities, sociological, normative, and technological which strictly interact. Some functional and non-functional requirements of the e-voting prototype, for instance, were built with a strict round-trip between the sociological and the technological line, with the normative line ensuring compatibility with the laws. See [11, 12] for more details and [13] for some considerations related to the sociological aspects of e-voting.

Three pilots have been conducted, so far, within the project, during local elections, in parallel to the standard procedures, with e-voting machines installed in the polling stations and with people participating on a voluntary basis. The first pilot was conducted May 2005 in four different towns of Trentino (Baselga di Piné, Fondo, Coredo, Lomaso), and in eight polling stations of the city of Trento. About seven thousand voters (55% of the people who actually voted at the election) took part to the trial. The second experimentation was conducted

in Daiano, in November 2005[1] About 330 voters tried the system, 90% of the people who actually voted at the election. The third trial was conducted last May in Peio, it involved about 1100 voters, with a participation of about 85%. With respect both to scope, population, and participation ProVotE is among the largest, if not the largest, e-voting project in Italy.

## 4 TheProVotE e-voting System: System Architecture

Even though the solution tested in trials involves various subsystems (to cover steps 1-5), here we focus on the development of the e-voting machine, since it is the most critical and "visible" component. The e-voting machine in fact has to be continuously available during the electoral day (6.00am to 10.00pm in Italy), it has to store votes safely and securely, it is used by all the voters and, as mentioned earlier, it is one of the main systems on which citizens judge the feasibility of e-voting.

**Fig. 1.** The ProVotE Machine

The ProVotE e-voting machine we developed (see Figure 1) is a DRE with a voter verifiable printed trail as suggested by e.g. [14],[2]. The voting machine is installed in a standard voting cabin. All the interaction happens through a touchscreeen which, during voting, reproduces the paper ballot. Votes are printed and shown to the voters, who are required to confirm them or make them void. Verification by the voter increases auditability of the system. The printer tape is cut after each vote (and the electronic data is *shuffled* after each vote), so that no sequence remains[2].

The machine is locked by default. A smartcard reader, outside the cabin, is used to unlock it. Two smartcards, that need to be alternated, are used to enable the machine for voting. This helps avoiding the possibility of casting more than one vote per voter. A third smartcard is used for the functions of the precinct's officers (e.g. opening and closing elections).

[1] Daiano was the only municipality in Trentino holding elections in November 2005.

[2] Being able to determine the order in which votes are given has resulted to be one of the concerns of Trentino's voters.

A set of lights (two green and one red) is used to inform about the state of voting machine. The red light signals that the machine is being used for voting and the green ones encode which card will enable the machine for the next vote.

The software system is based on a stripped-down version of Linux and an application (*jprovote*) written in Java (following e.g. [5]). The *jprovote* system is structured in four layers:

- **services,** which provides the basic functionality to the rest of the application, such as drivers for controlling the three-light indicator and the printer, managing logs for audits, and transparently managing redundant and ciphered persistence of data;
- **data model management,** which manages all the election specific data, comprising candidates and parties, the ballot data, per-machine election results, and the symmetric and asymmetric keys used for ciphering and signing;
- **control logic,** which defines how the machine has to react to user actions both in administration and in voting mode. The control logic also specifies the logic of the user interface (e.g. what screens has to be shown next).
- **user interface,** which manages the graphical layout of the administration and of the voting interface.

## 5 The e-voting Control Logic and its Development

The control logic component of *jprovote* is among the most critical part of the system. The component, in fact, is responsible of implementing a well-defined life cycle, which encodes the procedures defined by the law and which limits the operations poll-officers can do. Errors in its specification may lead to violations of fundamental principles on which the law is based (e.g. allowing a voter to vote twice, denying the right to vote to a voter). Finally, the e-voting machines need to be adaptable to different kind of elections and come in different configurations (e.g. testing, voting), which require both flexibility in the specification of the user interface and efficiency in being able to adapt the behavior of the control logic.

We decided, therefore, to specify the control logic using finite state machines. The architecture is shown in Figure 3. An *Event Queue Manager* handles all asynchronous events (e.g. smart-card inserted/extracted events, power fail events from the UPS) and feeds the *State Machine*, which reacts to the events and sends commands to a *Delegate*. The *Delegate*'s methods can be thought like atomic building blocks (e.g. 'print the current ballot', 'clear the interface', 'turn the outside red light on', etc.) that are combined by means of the statechart specification.

Particular care has also been put to the specification of this component. Broadly speaking, the process we followed is a waterfall which we adapted to incorporate (formal) verification activities. It is summarized in Figure 2:

- the initial requirements definition activity produced a requirements document which contains high-level statecharts and usecases. The statechart specifies the life cycle of the machine (e.g. opening the poll-site, starting

the counting, etc.) and the usecases provide the usage scenarios for each of the states of the machine's lifecycle (e.g. the actions necessary to open the poll-site).
The voting user interface is specified through a statechart which describes its logic and in which each state corresponds to a different "screenshots".

- the specifications and, in particular, the UML statecharts have been discussed and validated by the Electoral Service of the Province of Trento. The voting interface was validated using throw-away prototypes developed from the statecharts, in experiments set up by the Faculty of Sociology with selected voters before the trials.
- during the design phase, the statecharts validated by the Electoral Service have been detailed into an *executable* statechart, which implements[3] the control logic. During the specification phase, we stick to a very strict and simple subset of the statechart formalism, avoiding some of the semantic issues related to some constructs.
- the *executable* statechart specification was then translated using custom tools into the NuSMV input language, formally verified using model checking techniques, and automatically translated into the Java code implementing the control logic. For the validation, we manually specified the main principles/properties that the machine had to satisfy in CTL (see, e.g., [15] for a list of the high-level properties). In order to standardize their representation, we used the patterns found [16].
- the generated Java code was finally inspected by hand, to mitigate risks related to bugs in the translator, and the source code then compiled and packaged to produce *jprovote*. The application was finally verified during the System Test phase, conducted using standard techniques.

Even though the process followed above does not guarantee that the system is error-free (due to, e.g., completeness of properties, errors in the translators), the use of graphical notations and the formal verification of the control logic allowed to significantly increase our confidence in the conceptual correctness of the design of the control logic. The use of UML, in fact, significantly simplified interaction with other stakeholders and increased understandability of the requirements by the people in the electoral service. (The Electoral Service is now evaluating the possibility of making the activity diagrams used to specify the electoral processes available to the general public.)

In order to support effectively the development process described above various activities have been carried out with tools which, in some cases, we specifically adapted and developed for the project (see Figure 4):

- **UML Editor** All the diagrams have been written using ArgoUML [17], which supports guards with arguments in statecharts. The specifications of the state machines have been serialized to XMI [18];

[3] Here the difference between "specification" and "implementation" is irrelevant, as the statechart is used to generate Java code.

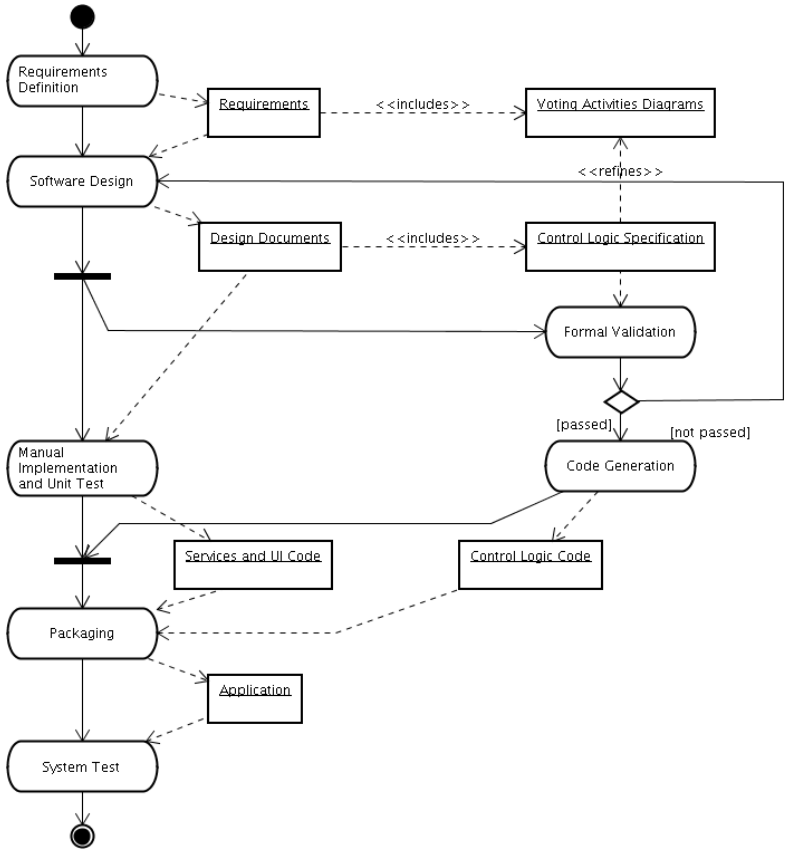


**Fig. 2.** ProVotE Development Process

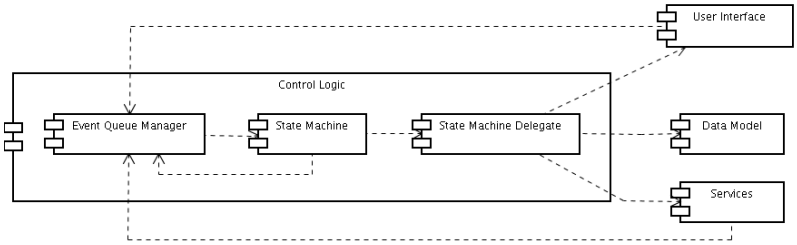


**Fig. 3.** Control Logic Component Structure

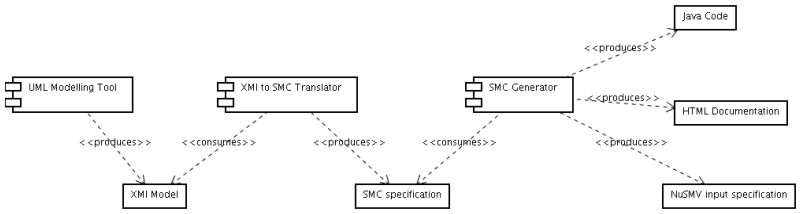


**Fig. 4.** Specification Validation and Code Generation Tool

– **FMSC+** which is a two-step compiler we develop from XMI to different output formats (Java, NuSMV input language, and HTML, for documentation purposes).
The FSMC+ front-end, which is based on JMI [19], reads an XMI specification and transforms it into the SMC input language. SMC (State Machine Compiler) is a open-source translator from a textual representation of a state machine to various output formats [20] (Java, C, C++, HTML, etc.) [20].
The FSMC+ back-end, is a rewritten version of the SMC tool that takes as input a SMC specification and produces various outputs (Java, NuSMV, HTML). The emitter is based on Velocity [21], a template engine written in Java.

See [22] for more details.

## 6 Related Work

While the development environment described above is similar to [23] and our tools and its design principles follow the ones presented in [24], there are some distinguishing features, among which:

– we support some features usually not available by other similar tools (e.g. UniMod [25], Tabu [26], Hugo [24]) such as event parameters, guards expression evaluated both on event parameters and delegate internal state;
– easier customization: code generation (or the generation of any other kind of content) is performed by means of a template engine. This allows to easily modify the generated content, e.g. to new languages or different coding styles;
– open source and modularity: the tools are based on open-source components and some tools (most notably the UML modeling tool), can be replaced with other tools providing XMI export;

As mentioned earlier, our approach supports only a subset of UML Statecharts. The goal, however, is not that of producing a general purpose tools, but, rather, simplifying the development of the e-voting system.

## 7 Conclusions and Future Work

This paper describes some of the technological activities we have been carrying out within the ProVotE project, which has the goal of introducing e-voting systems for the next provincial elections.

We believe that a transition to new technologies, especially in a country which is particularly cautious towards new technologies in the polling stations requires a multi-disciplinary approach that allows to take into account not only the usability requirements of the voters, but also those non-functional requirements that help guaranteeing security and build trust on the new voting machines.

So far, we tested our prototypes in pilots that are among the largest e-voting tests ever performed in Italy. Several changes and refinements still need to be implemented in the e-voting solutions, both functional (like audio interfaces for visually impaired people) and non functional, in order to reduce costs, size, and improve robustness of the prototypes.

The technological actions described above, together with the sociological, communication, and normative actions planned for the second phase will gradually broaden the size of experimentations to the whole province, allowing for a smooth introduction of e-voting systems in the province of Trento.

## 8 Acknowledgements

The work described in this paper has been and is being developed within the ProVotE Project, a project sponsored by Provincia autonoma di Trento.

As with any large project, the results presented in this paper based on the joint and coordinated work of several people. We wish in particular to thank: Sergio Bettotti, Carlo Buzzi, Letizia Caporusso, Alessandro Ceschi, Costantino Charalabopoulos, Giuseppe Conti, Claudio Covelli, Raffaele Deamicis, Giorgia Fasanelli, Alberto Faustini, Giolo Fele, Patrizia Gentile, Luciana Giuliani, Francesca Gleria, Bianca Lanzalone, Antonella Lavarian, Giuseppe Negri, Enrico Parola, Ernesto Passante, Giampaolo Pedrotti, Pierangelo Peri, Nicola Prantil, Roberto Resoli, Francesca Sartori, Gianfranco Stellucci, Catherine Tonini, Paolo Troebinger, Michele Welponer.

## References

1. David Chaum. Secret-ballot receipts: True voter-verifiable elections, 2004.
2. R. Mercuri. Explanation of voter-verified ballot systems. *ACM Software Engineering Notes (SIGSOFT)*, 27(5). Also at `http://catless.ncl.ac.uk/Risks/22.17.html`.
3. A. PRosser, R. Kofler, R. Krimmer, and M. K. Unger. Security assets in e-voting. In *the International Workshop on Electronic Voting in Europe*, 2004.
4. B. Var Acker. Remote e-voting and coercion: a risk-assessment model and solutions. In *the International Workshop on Electronic Voting in Europe*, 2004.
5. T. Kohno, A. Stubblefield, A.D. Rubin, and D.S. Wallach. Analysis of an electronic voting system, 2004.

6. Alessandro Cimatti, Edmund M. Clarke, Fausto Giunchiglia, and Marco Roveri. NUSMV: A new symbolic model checker. *International Journal on Software Tools for Technology Transfer*, 2(4):410–425, 2000.
7. Governo Italiano. *European Elections 2004, Automated Counting of the Votes (Elezioni Europee 2004, Conteggio Automatizzato del Voto)*. In Italian. Available at `http://www.governo.it/GovernoInforma/Dossier/voto_conteggio`.
8. MET Informatica Comune di San Benedetto del Tronto. Preliminar report on the electronic voting expreimentation (sperimentazione sulla votazione elettronica - preliminare). In Italian. Available at `http://www.comune.san-benedetto-del-tronto.ap.it/ePoll/rl00.html`.
9. E-Poll. Electronic polling system for remote voting operations. Available at `http://www.e-poll-project.net/`.
10. M. McGaley and J. McCarthy. Transparency and e-voting: Democratic vs. commercial interests. In *the International Workshop on Electronic Voting in Europe*, 2004.
11. Adolfo Villafiorita and Giorgia Fasanelli. Transitioning to evoting: the provote project and trentino's experience. 2006. To appear in the Proceedings of EGOV-06.
12. Letizia Caporusso, Carlo Buzzi, Giolo Fele, Pierangelo Peri, and Francesca Sartori. Transitioning to evoting and citizen participation, 2006. To appear in the Proceedings of eVoting-2006.
13. A.M. Ostveen and P. van den Besselaar. Security as belief - user's perceptions on the security of electronic voting systems. In *the International Workshop on Electronic Voting in Europe*, 2004.
14. R. Mercuri. A better ballot box? *IEEE Spectrum Online*, October 2002. Available on line at `http://www.spectrum.ieee.org/WEBONLY/publicfeature/oct02/evot.html`.
15. M. McGaley and J.P. Gibson. Electronic voting: A safety critical system. Final year project report, 2003.
16. M. B. Dwyer, G. S. Avrunin, and J. C. Corbett. Patterns in property specifications for finite-state verification. Technical Report UM-CS-1998-035, , 1998.
17. Tigris.org: Open Source Software Engineering Tools. Argouml. Available at `http://argouml.tigris.org/`.
18. OMG. Omg xml metadata interchange (xmi) specification.
19. Ravi Dirckze (Spec. Leader Unisys Corporation). Jsr-040 the javatm metadata interface (jmi) specification, 2002. Final Specification. Version 1.0.
20. Charles W. Rapp. The state machine compiler. http://smc.sourceforge.net/.
21. The Apache Jakarta Project. Velocity. Available at `http://jakarta.apache.org/velocity`.
22. Silvia Tomasi. Metodi formali nello sviluppo di applicazioni per il voto elettronico: l'esperienze nel progetto provote. Master's thesis, University of Trento, 2006. In Italian.
23. Adm Darvas and Istvn Majzik. Verification of uml statechart models of embedded systems.
24. Timm Schafer, Alexander Knapp, and Stephan Merz. Model checking uml state machines and collaborations. 55, 2001.
25. Unimod. Available at `http://unimod.sourceforge.net`.
26. M. Encarnacin Beato, Manuel Barrio-Solrzano, and Carlos E. Cuesta. Uml automatic verification tool (tabu), 2004.