

Ensuring reliability in electronic examinations through UPPAAL-based trustworthy design

The paper is not well written and the presentation should be improved in several ways. In addition, there are a few open questions that authors should be addressed to improve the work. Please observe to the following detailed review of each section:

Preliminaries:

In this section, the authors are missing several concepts of the problem they are trying to analyze. There are several open questions on the simplification they are presenting. For instance, there are only multiple students but only one examiner and invigilator and what is the process to finish an examination? In addition, the problem simplification authors are considering is not analyzed to determine if this is an appropriate starting point for further analysis of this problem. Although, as an expert in the field, this approach is very simplistic. Please, use the notation of UML sequence chart this is a standard notation everybody can understand to avoid misunderstandings.

When UPPAAL is presented in this section, the introduction is sparse in important details. They are using this tool to specify the behavior of a system, but important details are missing, for instance, the concept of state and the management of time. Besides, there are important errors from the theoretical point of view when certain elements are presented and many of them are missing, for instance, variables and the management of information are not introduced, either.

Model specification:

Candidate template:

Authors refer to variable `j` to manage the current question but, in this template, there is a structure `q_cnt` to manage the current question instead of `j`.

When there is a submission, the synchronization uses `Submit[i]!` and then the function `SetA(i,i)` is executed. This action establishes the answer to the current question using the same identifier of the candidate. This isn't very clear. I can understand you are not

managing real information, but you should give a proper explanation in the paper or you should consider another approach. In addition this function should be explained:

```
//set a current answer of candidate i
```

```
void SetA(ID i, QS a){  
    cand[i].item[q_cnt[i]].a = a;  
}
```

Other open questions in this template. You are now introducing how the process ends. It should be explained previously in the preliminaries.

The concept of broadcast channel is used for reset, but it is not introduced here:

```
broadcast chan reset;
```

In addition, this process has not been explained in the preliminaries.

Administration template

The description is quite cryptic and refers to information that has not been properly introduced as information management using queues. This is quite important to understand that this paper is not well presented, and it is difficult to understand.

Authors should introduce what are the queues used and what are they encoding:

```
//T: Total, R: Register, S: Submit, A: Accept
```

```
OpQueue T, R, S, A;
```

The function EnOpQueue has not been introduced again.

```
//Add an element to an OpQueue
```

```
bool EnOpQueue(OpQueue& q, int xi, int xop, NCQS xq, int xa)
```

In addition, all the operations codes has not been explained and the text is hard to follow:

OpType op; //operation type: -1: none, 0: start, 1: register, 2: login, 3: get, 4: submit, 5: accept, 6: corrAns, 7: mark, 8: notify, 9: end, 10: reset

Neither, the operation findelement has been introduced...

Invigilator template

Authors should explain the use of the global variable isa. This kind of information should be used when the information needs to be share with other, which I do not think is the case. You can use a local variable to this template instead.

In addition, the time passage is not considered here. The invigilator can remain in the submit state indefinitely, but this is not explained. This problem can cause that t clock evolves to $t \geq 100$ a guard considered in the edge between Process and End. By the way, why are you using here this guard ($t \geq 100$), when you are applying a TMax of 1000? I don't understand it.

Examiner template

Authors are not explaining the uses of functions: CorrAns, MarkScore, ComputeSMatrix.

```
//set correct answers ca[Q] to questions
```

```
void CorrAns(QS& ca[Q]){
```

```
    int i;
```

```
    for(i = 0; i < Q; i++)
```

```
        ca[i] = 0;
```

```
}
```

This function sets all valid answers to 0 and later in the submission you are defining the answers in the candidate templates using its identifier. Therefore, candidate with id 0 gets all the answers right and candidate with id 1 gets all the answers wrong. Is this assumption right? You have not introduced this information.

```
//mark socres and the total score for every candidate
```

```

void MarkScore(CandidateType& cand[N], QS& ca[Q]){
    int i, j, k;
    for(i = 0; i < N; i++)
    {
        cand[i].total = 0;
        for(j = 0; j < Q; j++)
        {
            if(cand[i].item[j].a == ca[j])
                cand[i].item[j].s = 1;
            else
                cand[i].item[j].s = 0;
            cand[i].total = cand[i].total + cand[i].item[j].s;
        }
    }
}

```

Authors uses variables ca and cand when these functions are called but they have not been introduced first.

I have no clue what are you using this function for:

```

//compute Similarity Matrix
void ComputeSMatrix(CandidateType& cand[N], QS& ca[Q]){
    int i, j, k;
    for(i = 0; i < N; i++)
        for(j = i + 1; j < N; j++)
            for(k = 0; k < Q; k++)
                if(cand[i].item[k].a == cand[j].item[k].a)
                    sm[i][j]++;
}

```

System declaration

When the system is declared in the paper, it is stated that only one candidate is considered, but observing the downloaded xml file provided in the text authors considers two candidates instead:

```
// Place template instantiations here.
```

```
C0 = Candidate(0);
```

```
C1 = Candidate(1);
```

```
AD = Administrator();
```

```
I = Invigilator();
```

```
E = Examiner();
```

```
// List one or more processes to be composed into a system.
```

```
system C0, C1, AD, I, E;
```

See, they are using C0 and C1.

Property specifications

Authors state that the set of properties checked ensure reliability and security, but there is not a detailed analysis of these properties first.

(5) Answer singularity

The following function has not been introduced. The legibility of it is not clear.

```
bool OneAnswerEachQuestion(OpQueue q, int xi)
```

```
{
```

```
    int xq;
```

```

int i, j;
for(xq = 1; xq < Q + 1; xq++)
    for(i = q.front; i < q.rear; i++)
        for(j = i + 1; j < q.rear; j++)
            if(q.data[i].id == xi && q.data[j].id == xi && q.data[i].q == xq && q.data[j].q ==
xq)
                return false;
return true;
}

```

(6) Acceptance assurance

Again the same problema with the function `FirstSubmitFollowAccept`

(7) Question ordering

Again the same problema with the function *GetAcceptGet*

The problem continues through the section. Even the concept of cheater is even introduced which I don't really think it captures this behavior properly using a distance algorithm.

In addition, in the file provided other properties are introduced. They number three times more than the 12 considered here. Are they necessary, then why are they have not been introduced.

Related work

The comparison here is only partial with other related work. Authors have not analyzed the differences in detail and determine if research results are different or similar.

For instance:

Kassem et al. proposed an event-based model of e-exams, defined several security properties
464 and validated these properties by analyzing real e-exams at UGA using ProVerif
and MarQ
465 (Kassem et al., 2017). However, the explicit interactions between roles in electronic
exams are
466 not taken into consideration. In their models, two methods are employed, namely
ProVerif and
467 QEA (Quantified Event Automata). We consistently utilize the UPPAAL
automaton to model all
468 these processes.
469 Bella et al. proposed

Authors only analyze the use of a different formal tool, but they do not perform an
analysis of the results obtained in the work.

Please, see the comments stated in the article as well.

Ensuring reliability in electronic examinations through UPPAAL-based trustworthy design (#99257)

1

First submission

Guidance from your Editor

Please submit by **14 May 2024** for the benefit of the authors (and your token reward) .



Structure and Criteria

Please read the 'Structure and Criteria' page for general guidance.



Raw data check

Review the raw data.



Image check

Check that figures and images have not been inappropriately manipulated.

If this article is published your review will be made public. You can choose whether to sign your review. If uploading a PDF please remove any identifiable information (if you want to remain anonymous).

Files

Download and review all files from the [materials page](#).

7 Figure file(s)

2 Table file(s)




Structure and Criteria

Structure your review

The review form is divided into 5 sections. Please consider these when composing your review:

1. BASIC REPORTING
2. EXPERIMENTAL DESIGN
3. VALIDITY OF THE FINDINGS
4. General comments
5. Confidential notes to the editor






 You can also annotate this PDF and upload it as part of your review

When ready [submit online](#).





Editorial Criteria

Use these criteria points to structure your review. The full detailed editorial criteria is on your [guidance page](#).




BASIC REPORTING

-  Clear, unambiguous, professional English language used throughout.
-  Intro & background to show context. Literature well referenced & relevant.
-  Structure conforms to [PeerJ standards](#), discipline norm, or improved for clarity.
-  Figures are relevant, high quality, well labelled & described.
-  Raw data supplied (see [PeerJ policy](#)).

EXPERIMENTAL DESIGN

-  Original primary research within [Scope of the journal](#).
-  Research question well defined, relevant & meaningful. It is stated how the research fills an identified knowledge gap.
-  Rigorous investigation performed to a high technical & ethical standard.
-  Methods described with sufficient detail & information to replicate.

VALIDITY OF THE FINDINGS

-  Impact and novelty not assessed. *Meaningful* replication encouraged where rationale & benefit to literature is clearly stated.
-  All underlying data have been provided; they are robust, statistically sound, & controlled.
-  Conclusions are well stated, linked to original research question & limited to supporting results.



The best reviewers use these techniques

Tip

Example

Support criticisms with evidence from the text or from other sources

Smith et al (J of Methodology, 2005, V3, pp 123) have shown that the analysis you use in Lines 241-250 is not the most appropriate for this situation. Please explain why you used this method.

Give specific suggestions on how to improve the manuscript

Your introduction needs more detail. I suggest that you improve the description at lines 57- 86 to provide more justification for your study (specifically, you should expand upon the knowledge gap being filled).

Comment on language and grammar issues

The English language should be improved to ensure that an international audience can clearly understand your text. Some examples where the language could be improved include lines 23, 77, 121, 128 – the current phrasing makes comprehension difficult. I suggest you have a colleague who is proficient in English and familiar with the subject matter review your manuscript, or contact a professional editing service.

Organize by importance of the issues, and number your points

1. Your most important issue
2. The next most important item
3. ...
4. The least important points

Please provide constructive criticism, and avoid personal opinions

I thank you for providing the raw data, however your supplemental files need more descriptive metadata identifiers to be useful to future readers. Although your results are compelling, the data analysis should be improved in the following ways: AA, BB, CC

Comment on strengths (as well as weaknesses) of the manuscript

I commend the authors for their extensive data set, compiled over many years of detailed fieldwork. In addition, the manuscript is clearly written in professional, unambiguous language. If there is a weakness, it is in the statistical analysis (as I have noted above) which should be improved upon before Acceptance.

Ensuring reliability in electronic examinations through UPPAAL-based trustworthy design

Wenbo Zhou^{1, 2, 3}, Yujiao Zhao¹, Ye Zhang¹, Liwen Mu^{1, 4}, Yiyuan Wang^{Corresp., 1, 3}, Minghao Yin^{Corresp. 1, 3}

¹ School of Information Science and Technology, Northeast Normal University, Changchun, Jilin Province, China

² Guangxi Key Lab of Multi-source Information Mining & Security, Guangxi Normal University, Guilin, Guangxi Province, China

³ Key Laboratory of Applied Statistics of Ministry of Education, Northeast Normal University, Changchun, Jilin Province, China

⁴ School of Physics, Northeast Normal University, Changchun, Jilin Province, China

Corresponding Authors: Yiyuan Wang, Minghao Yin

Email address: wangyy912@nenu.edu.cn, ymh@nenu.edu.cn

Electronic examination serves as an efficient method for assessing learning outcomes, yet the integration of computers into exam processes introduces potential for unreliability. In this paper, we propose a formal model for electronic examinations using timed automata, providing a structured approach to understanding and managing the complexities. The electronic examination process is modeled by defining four UPPAAL templates, i.e., candidate, administrator, invigilator, and examiner. Crucial properties specific to electronic examination are encoded as specifications in UPPAAL. Verification against these properties demonstrates the validity and reliability of this model. The modelable and verifiable electronic examination designed with UPPAAL suggests great potential for deeper exploration in trustworthy digital education.

Ensuring reliability in electronic examinations through UPPAAL-based trustworthy design

Wenbo Zhou^{1,2,3}, Yujiao Zhao¹, Ye Zhang¹, Liwen Mu^{1,4}, Yiyuan Wang^{1,3} and Minghao Yin^{1,3}

¹ School of Information Science and Technology, Northeast Normal University, Changchun, Jilin Province, China

² Guangxi Key Lab of Multi-source Information Mining & Security, Guangxi Normal University, Guilin, Guangxi Province, China

³ Key Laboratory of Applied Statistics of Ministry of Education, Northeast Normal University, Changchun, Jilin Province, China

⁴ School of Physics, Northeast Normal University, Changchun, Jilin Province, China

Corresponding Author:

Yiyuan Wang^{1,3} and Minghao Yin^{1,3}

2555 Jingyue Street, Changchun, Jilin Province, 130117, China

Email address: wangyy912@nenu.edu.cn, ymh@nenu.edu.cn

Abstract

Electronic examination serves as an efficient method for assessing learning outcomes, yet the integration of computers into exam processes introduces potential for unreliability. In this paper, we propose a formal model for electronic examinations using timed automata, providing a structured approach to understanding and managing the complexities. The electronic examination process is modeled by defining four UPPAAL templates, i.e., candidate, administrator, invigilator, and examiner. Crucial properties specific to electronic examination are encoded as specifications in UPPAAL. Verification against these properties demonstrates the validity and reliability of this model. The modelable and verifiable electronic examination designed with UPPAAL suggests great potential for deeper exploration in trustworthy digital education.

Introduction

Examinations play a preeminent pedagogical role, enabling individuals to assess their skills and knowledge in a specific subject (Giustolisi, 2018). In contemporary education, e-learning has emerged as a prevalent option for universities, facilitating the convenient expansion of teaching or learning activities at any time and from anywhere. Particularly with the outbreak of the epidemic, e-learning has garnered increased attention in the field of education. As a vital component of e-learning, electronic examinations serve an irreplaceable function in facilitating efficient evaluation of learning outcomes.

Electronic examinations offer a convenient means to assess learning outcomes utilizing ubiquitous internet-connected devices. The adoption of online learning management systems has enabled universities to boost enrollments without the need for additional lecture halls (Ngqondi et al., 2021). However, electronic examinations have been met with skepticism, primarily due to concerns about academic fraud (Ngqondi et al., 2021). Introducing computers into various phases of examinations brings new security challenges (Bella et al., 2017). Simultaneously, many learners express apprehensions about potential technical glitches during the implementation of electronic examinations (Ilgaz and Afacan Adanır, 2020). In such a scenario, ensuring the reliable, fair, and seamless execution of electronic examinations in e-learning becomes paramount (Muzaffar et al., 2021). The design of electronic examinations should prioritize reliability with due consideration to its magnitude.

Reliability in the context of electronic examinations is pivotal to ensure the seamless execution of exams and uphold fairness. In the realm of education, examination rules are explicitly outlined, mandating strict adherence artificially. For electronic examinations, it becomes imperative for such rules to seamlessly integrate with examination systems. Correct-by-construction is among the most effective approaches to guarantee the reliability of software systems right from the source. Formal methods provide robust assurances during system design, ensuring the strict adherence to examination rules. These methods typically involve constructing suitable formal models and verifying them before proceeding to further implementation. Adopting this methodology offers numerous advantages. On one hand, it allows for the identification and correction of errors at an early stage, preventing them from evolving into intricate and challenging faults that are both costly and time-consuming to diagnose and repair. On the other hand, formal methods employ a range of proving or model-checking techniques. Notably, executable models can be directly simulated and analyzed, enabling the exploration of diverse design options and verification possibilities (Bobbà et al., 2018).

To construct an abstract and executable model, the chosen formal method must be appropriate and highly applicable. As one of the most widely employed formal methods, timed automata are a valuable formalism for verifying concurrent systems under timing constraints (Arcile and André, 2022). It has been extensively applied to modeling and analysis of many critical systems as well as assistance in the generation of new methods (Hofmann and Schupp, 2023, Lehmann and Schupp, 2022), including various industrial applications (Basile et al., 2021, Sakata et al., 2023). One of the most significant advantages of time automata lies in their comprehensibility through graphical representation. Another strength is their executability with timing constraints, enabling the exploration of diverse scenarios and the validation of critical properties. Furthermore, well-established support tools like UPPAAL, contribute substantial capabilities to practical modeling, simulation, and verification. Therefore, we employ UPPAAL to model and analyze the electronic examination system.

In this paper, we introduce a model for an electronic examination system based on timed automata. We conduct simulations and analyses of crucial properties using the UPPAAL tool. The UPPAAL model we developed is available in this site: <https://github.com/TURTING->

BO/An-Electronic-Examination-Model-Based-on-UPPAAL. To the best of our knowledge, this is the initial attempt to model examinations using UPPAAL. By leveraging the modeling and verifiable capabilities of UPPAAL, there exists the potential for assessing and enhancing electronic examinations in an understandable manner. This electronic examination model can function as a foundational component, offering more possibility for further exploration in the realm of trustworthy digital education. To save space, we assume the reader has a fundamental understanding of the UPPAAL tool. More detailed definitions regarding the syntax and semantics of languages used in UPPAAL can be referenced in (UPPAAL Team, 2023). We note that cryptography aspects are not within the scope of this paper. The contributions of this paper are outlined as follows:

- A formal model for electronic examination presented based timed automata.
- Typical properties related to electronic examination are encoded as specifications in UPPAAL.
- Our model is verified against these properties using UPPAAL to demonstrate the reliability of electronic examination.

The remainder of this paper is structured as follows. Section 2 provides a brief overview of the background on electronic examinations and UPPAAL. Section 3 details the model specification and property specification for electronic examination. In Section 4, we validate and verifies typical properties using UPPAAL. Section 5 discusses related work. Finally, Section 6 concludes this paper and outlines future work.

Preliminaries

In this section, fundamental concepts about electronic examinations and UPPAAL are introduced to enhance comprehension of the following models. Regarding electronic examinations, we illustrate the roles of participants and their interactions. As for UPPAAL, a concise description of its core components is provided.

Electronic examination

Electronic examinations offer a convenient way for assessing the knowledge and abilities of learners with the help of computer and network technologies. The participants in electronic examinations are akin to those in traditional exams, with the difference that certain operations are conducted through network communications.

Roles of participants

In a general way, there are four roles in an electronic examination, i.e., candidate, administrator, invigilator and examiner. We list their functions as follows.

- *Candidate*: A candidate is a student taking the examination.
- *Administrator*: An administrator is responsible for registering candidates for the examination.
- *Invigilator*: An invigilator is tasked with distributing questions, supervising the examination and collects answers.
- *Examiner*: A examiner marks the examination and notifies students their scores.

Each participant is assigned specific tasks, and through collaborative efforts, they contribute to the successful execution of electronic examinations.

Interactions among participants

This subsection explores the dynamic interactions that take place among the various participants involved in the electronic examination process. As depicted in Fig. 1, communication among these participants occurs through various operations. Prior to the commencement of an electronic examination, the examiner must establish correct answers for the questions (corrAns). The sequence unfolds as follows: First, a candidate, denoted as c_i , registers with the administrator (register). Then, c_i logs into the examination system and notifies the invigilator (login). Next, the invigilator dispatches a question to c_i (get), who, in turn, formulates and submits an answer (submit). Upon receiving the answer, the invigilator confirms the submission through acknowledgment (accept). This iterative process may repeat multiple times based on the number of questions. Finally, the invigilator notifies the examiner to commence grading (mark) and informs c_i of their scores (notify).

UPPAAL

UPPAAL is a real-time system model checker based on timed automata, developed collaboratively by Uppsala University and Aalborg University. UPPAAL comprises an editor, two simulators, and a verifier. The functions of the editor, simulator, and verifier are briefly introduced as follows.

Editor

The editor is utilized for system modeling, involving the construction of a network of timed automata. A system model primarily includes declarations for global variables, templates (comprising local variable declarations), and system declarations. Global variables encompass synchronization variables, clock variables, and other relevant variables. Each template corresponds to a timed automaton and serves as a model specification, as detailed below. System declarations instantiate templates, culminating in the creation of a network of timed automata.

Model specification. The target system is represented as a network of timed automata using UPPAAL's graphic editor. Types of timed automata are established as templates in UPPAAL, with each template capable of instantiation as a series of automata of the same type. A template primarily consists of two elements: locations and edges (Uppsala University and Aalborg University, 2023).

- *Location.* A location represents a state within the system. There are three types of locations, i.e., initial location, urgent location and committed location. An *initial location* indicates the initial state of a timed automaton. An *urgent location* implies that time does not pass while in this location. A *committed location* also halts the passage of time; however, the distinction lies in the requirement that the next triggered transition must be an outgoing edge from any committed location.
- *Edge.* An edge connects two locations (or itself), representing the transition between two states. It involves four types of labels: select, guard, synchronization, and update. *Select* denotes the random selection of a value within a range. *Guard* is a Boolean expression used

to determine the enabling of a transition. *Synchronization* is a label that synchronizes through two expressions $c!$ and $c?$, on the **channel** variable c . *Update* changes values of variables.

161 **Simulator**

The simulator serves the purpose of automatically generating and demonstrating execution paths. Two types of simulators are available: a symbolic simulator and a concrete simulator. The symbolic simulator dynamically executes and visually displays paths, while the concrete simulator enables users to set specific triggering time for transitions. These simulators offer a variety of operations, including *Reset*, *Next*, *Prev*, *Replay*, *Random*, *Shrink*, *Expand*, and *Speed Selection*. Within the graphic windows, state diagrams and message sequence charts are presented to enhance the understanding of simulations.

169 **Verifier**

The verifier extensively explores the state space corresponding to a system model, checking whether the specified properties are satisfied. The properties are specified using the above query language.

Property specification. The query language primarily encompasses five types of path formulae, i.e., $E\langle\rangle p$, $A\langle\rangle p$, $E[]p$, $A[]p$, and $p \rightarrow q$, where p and q are **state formulae** such as $i == 4$. $E\langle\rangle p$ signifies there exists a state in a path satisfies p . $A\langle\rangle p$ indicates there exists a state in every path satisfies p . $E[]p$ denotes all states in a path satisfies p . $A[]p$ signifies all states in all paths satisfy p . Finally, $p \rightarrow q$ means that whenever p is satisfied, q will also be satisfied.

In UPPAAL, the query language is a subset of timed computation tree logic (TCTL) language. Once users define property specifications and click the *Check* button, the verifier assesses whether the system model satisfies each property.

182 **Modeling electronic examination using UPPAAL**

In this section, we provide a detailed introduction to the notation, model specification, and property specification related to electronic examinations.

185 **Notation**

There are primarily three data structures: *Operation*, *Item*, and *TotalScore*. We present their definitions as follows. An operation is a quadruple (id, op, q, a) where:

- 188 (1) id represents the identification of a candidate, where $id \in \mathbb{Z}$;
- 189 (2) op denotes an operation label, where $op \in \{start, register, login, get, submit, \text{accept},$
190 $corrAns, mark, notify, \text{end}, reset\}$;
- 191 (3) q indicates a question;
- 192 (4) a indicates an answer.

An operation captures the details of an action performed by a role, encompassing four types of information: candidate identification, operation label, question, and answer. Each candidate possesses a unique identification. Ten operations are considered to facilitate interactions among different roles. The *start*, *mark*, *end*, and *reset* labels signify distinct phases in an electronic examination. For instance, the *mark* label indicates that the examination has concluded, and it is time for scoring. The *register* and *login* labels require only candidate identification information.

The *corrAns* label is used to set the correct answer for each question, necessitating information about the question. Similarly, the *notify* label requires candidate identification information. Finally, the *get*, *submit*, and *accept* labels are designated for handling actions such as receiving a question, submitting an answer, and accepting a submission. An example of a submission operation is provided in Example 3.1.

Example 3.1. A submission operation, denoted as $(cand_1, submit, q_2, a_2)$, signifies that the candidate $cand_1$ is submitting an answer a_2 in response to question q_2 .

In the UPPAAL implementation, when certain information is unnecessary, we set the corresponding value to -1. An example of a register operation is illustrated below.

Example 3.2. A correct answer setting operation is denoted as $(-1, corrAns, q_3, a_3)$. This operation indicates the correct answer for a question q_3 is set to a_3 .

For each question, the candidate submits a corresponding answer, and the examiner marks it according to the correct answer. We define an *item* as a tuple that associates a question with the candidate's answer and the score marked by the examiner. An item is a triple (q, a, s) where:

- (1) q represents a question;
- (2) a represents an answer;
- (3) s denotes the score corresponding to the question and the answer.

An *Item* exclusively records information about a single question. Consequently, we further define *CandScore* to collect all the items and calculate the total score for notifying a candidate. A candidate is a pair $(items, total)$ where:

- (1) *items* indicates a set of items;
- (2) *total* denotes the total score with respect to the items.

After defining the above concepts and encoding them into declarations in UPPAAL, we can construct a series of models for an electronic examination system using these data structures.

Model specification

In UPPAAL, the model specification is presented in the form of timed automata templates. In our model, there are four templates corresponding to the candidate, administrator, invigilator, and examiner, as follows.

Candidate template

A candidate is an individual taking the examination as shown in Figure 2. Initially, the candidate must register with the administrator using the `register[i]!` synchronization. Following a permission check, the candidate can log in to commence the examination. Simultaneously, the current question number, denoted as j , is initialized to 0. Subsequently, the candidate engages in a loop where they receive a question, submit their answer, and confirm the acceptance of the answer.

The `get[i]?` synchronization is used to acquire questions sequentially from the invigilator. Upon obtaining a question, the question number is recorded in *cand*, a data structure implementing a set of *Candidates*. After responding to the question, the candidate uploads the answer using the `submit[i]!` synchronization. To ensure acknowledgment of the answer by the invigilator, the candidate employs an `accept[i]?` synchronization.

When the number of answered questions equals or exceeds Q (a default parameter representing the total number of questions), the candidate transitions to the **Notify** location. Finally, upon receiving notification of their scores through the `notify[i]?` synchronization, the candidate reaches the **End** location, and the `end[i]` flag is set to true. Subsequently, all states are reset using the `reset?` synchronization.

Administrator template

The administrator is responsible for information management, encompassing initialization, correct answer setting, and registration information maintenance, as shown in Figure 3. Before any processes, the administrator automaton must initialize all necessary variables and reach the **Ready** location. Through the `corrAns?` synchronization, correct answers to questions are set. Subsequently, the administrator automaton moves to the **Start** location, signifying the commencement of the examination. Concurrently, a start operation, encoded as $(-1, 0, -1, -1)$, is added to the T queue.

When a candidate requests registration in the examination system, the administrator verifies whether the candidate has already registered. If not, the candidate's registration information is stored in the R queue. Additionally, a *register* operation, encoded as $(i, 1, -1, -1)$, is enqueued to the T queue. Finally, when the *AllEnd* function confirms that all automata have reached an **End** location, the administrator issues an instruction to reset all automata.

Invigilator template

An invigilator oversees the examination proceedings, taking on the responsibility of verifying candidates' logins, dispatching questions to candidates, receiving and confirming the submission of answers, as shown in Figure 4.

Initially, as candidates log in, the system transitions to the **Login** location. Once all candidates are prepared for the examination (i.e., the number of logged-in candidates equals N), the invigilator synchronizes with candidates using `get[i]!` and *SetQ* to transmit and update the questions. Simultaneously, a *get* operation, represented as $(i, 3, GetQ(i), GetA(i))$, is placed into the T queue.

Upon a candidate submitting an answer, the invigilator verifies their registration. If confirmed, the `submit[i]?` synchronization takes place, and a *submit* operation, encoded as $(i, 4, GetQ(i), GetA(i))$, is queued. The globally shared variable *isa* is employed to store the current candidate id. Subsequently, the automaton reaches the **Submit** location. Through the `accept[isa]!` synchronization, the invigilator notifies the candidate that their answer to the question has been received. Additionally, the *accept* operation is added to the T queue, and the question counter is updated.

Finally, if all candidates have responded to all questions, the invigilator automaton advances to the **End** location, notifies the examiner that the marking process can commence, and sets the *iend* flag to *true*.

Examiner template

An examiner is responsible for setting the correct answers to questions, evaluating candidates' responses, and communicating their scores to the candidates, as shown in Figure 5. Initially, the

examiner employs the *corrAns!* synchronization to provide the correct answers before the examination, and a *corrans* operation, represented as $(-1, 6, -1, -1)$, is inserted into the T queue.

Subsequently, upon receiving the synchronization signal *mark?*, the examiner assesses each candidate. Simultaneously, the answer similarity matrix is computed based on the *Candidate* information and the correct answers. A *mark* operation, encoded as $(-1, 7, -1, -1)$, is then added to the T queue.

Following the evaluation, the examiner uses the *notify[imn]!* synchronization to inform candidates of their scores. Finally, once all candidates have been notified of their scores (i.e., $imn \geq N$), the *eend* flag is set to *true*.

System declaration

Based on the four UPPAAL templates mentioned above, a comprehensive system can be generated by interconnecting these timed automata into a network. The system declaration is presented as follows.

system Candidate, Administrator, Invigilator, Examiner;

Property specifications

A trustworthy electronic examination system must adhere to a set of properties to ensure reliability and security. These properties can be articulated through formal specifications. According to the literature (Kassem et al., 2017), we consider the following twelve properties, which are encoded using a simplified version of TCTL, serving as property specifications in UPPAAL.

(1) No deadlock

In the electronic examination model, the absence of deadlocks is crucial to prevent any "never-ending" scenarios. Deadlocks, where processes wait indefinitely for each other, disrupt the system's flow. By implementing effective process synchronization and careful system design, we ensure a smooth examination experience, free from any prolonged or unresolved situations. Describing the absence of deadlocks in UPPAAL as a query is straightforward:

$A[] \text{ not deadlock}$

(2) Candidate registration

The candidate registration property stipulates that a candidate can submit an answer only if they have registered. This assertion is verified using two queues, namely R and S , dedicated to storing *register* and *submit* operations, respectively. The query is articulated as follows, where $\text{forall}(i:ID)$ denotes every candidate, and $\text{FindElement}(Q, i)$ is a function that assesses whether an operation with candidate identification i belongs to the Q queue. The candidate registration property underscores that submission is contingent upon prior registration, and this condition is rigorously examined through the R and S queues.

$A[] \text{ forall}(i:ID) \neg (! \text{FindElement}(R, i) \ \& \ \text{FindElement}(S, i))$

The query asserts that for every state within each path of the state space, there is no instance where an operation exists in the S queue but is absent in the R queue. In simpler terms, it implies that no candidate can submit without first registering.

(3) Candidate eligibility

The candidate eligibility property signifies that a candidate's answer can be accepted only if they have registered. To formulate this query, we utilize two queues R and A , specifically for the *register* and *accept* operations. The query is expressed as follows.

$$A[] \text{forall}(i:ID) \neg(\neg \text{FindElement}(R, i) \ \& \ \text{FindElement}(A, i))$$

In this query, for every state within each path of the state space, there is no instance where an operation exists in the A queue but is absent in the R queue. In other words, it asserts that if a candidate does not register, their answer cannot be accepted.

(4) Answer authentication

The answer authentication property stipulates that a candidate's answer can be accepted only if they have submitted the answer. To articulate this query, we utilize two queues for *submit* and *accept* operations, denoted as S and A respectively. The query is formulated as follows.

$$A[] \text{forall}(i:ID) \neg(\neg \text{FindElement}(S, i) \ \& \ \text{FindElement}(A, i))$$

This query asserts that, for every state within each path of the state space, there is no instance where an operation exists in the A queue but is absent in the S queue. In essence, it emphasizes that if a candidate does not submit an answer, that answer cannot be accepted.

(5) Answer singularity

The answer singularity property signifies that, for each candidate, only a singular response can be deemed acceptable per question. The function *OneAnswerEachQuestion* is designed to verify the presence of operations within queue A that share identical questions.

$$A[] \text{forall}(i:ID) \text{OneAnswerEachQuestion}(A, i)$$

The query checks each state within every path of the state space, ensuring that the *OneAnswerEachQuestion* function consistently yields true. This indicates that, for each question, only a single response is admissible from a given candidate.

(6) Acceptance assurance

The acceptance assurance property underscores the requirement that an answer submitted by a candidate should be accepted. In this context, emphasis is placed on the initial submission, implying that the first *submit* operation related to a question from a candidate is succeeded by an *accept* operation. The following *FirstSubmitFollowAccept* function describes this query.

$$A[] \text{forall}(i:ID) \text{FirstSubmitFollowAccept}(T, i)$$

In this query, for each state in every path of the state space, the *FirstSubmitFollowAccept* function consistently yields true. This implies that, after the first submission of an answer from a candidate, the invigilator accepts the answer.

(7) Question ordering

The question ordering property emphasizes that a candidate can proceed to the next question only after the answer to the current question is accepted. The *GetAcceptGet* function articulates this property by specifying that a *get* operation for question i is succeeded by an *accept* operation for the answer to question i . Furthermore, the *accept* operation is succeeded by a *get* operation for question $i+1$. It is important to note that the handling of the last question involves a special consideration.

$$A[] \text{forall}(i:ID) \text{GetAcceptGet}(T, i)$$

The aforementioned query signifies that, for each state in every path of the state space, the sequence of operations, namely $get(i)-accept(i)-get(i+1)$, remains unbroken. Consequently, this ensures that a candidate can systematically answer questions one after another.

(8) Exam availability

The exam availability property stipulates that the acceptance of an answer from a candidate is permissible only during the examination period. This implies the presence of an *accept* operation between the *start* operation and the *end* operation within the T queue. This property is conveyed through the *StartAcceptEnd* function in the following query.

$$A[] \text{ StartAcceptEnd}(T)$$

The query checks whether the *StartAcceptEnd* function is satisfied for every state in every path of the state space. Given the singular occurrence of both the *Start* and *End* operations, we abstractly take the sequence spanning from *Start* to *End* as the examination period.

(9) Answer-score integrity

The answer-score integrity property ensures that the correct answer can not be modified after the examination starts. This property is captured as the following *NoStartCorrAns* function in the query.

$$A[] \text{ NoStartCorrAns}(T)$$

This query guarantees that no *corrAns* operation is succeeded by *start* in the T queue, signifying that the correct answers can only be set before the commencement of an examination.

(10) Cheater detection.

During an examination process, cheating may take place, e.g., one candidate copies the answers of the other candidate. In this paper, we only consider the basic form of cheating, namely copying. The *NoCorrAnsMark* function is employed to verify this property.

$$A[] \text{ NoDistanceExceed}(sm)$$

To assess the answer similarity between two candidates, a matrix, denoted as $sm[i][j]$, is computed to measure the degree of similarity between candidate i and candidate j . If $sm[i][j] > k$ where k is a constant representing the tolerance for duplication, the *NoDistanceExceed* function returns *false*, indicating a potential cheating event. Here, the $sm[i][j]$ corresponds to the count of same answers between candidate i and candidate j .

(11) Marking correctness.

The marking correctness property asserts that once marking has occurred, the correct answers cannot be modified. This property is verified by the following query, employing a function named *NoCorrAnsMark*.

$$A[] \text{ NoCorrAnsMark}(T)$$

This query ensures that no *CorrectAns* operation is succeeded by the *Mark* operation in the T queue, preventing any modification of correct answers during the marking process.

(12) Mark integrity.

The mark integrity property ensures that each candidate receives notification after marking, and all answers from candidates are duly marked. The verification of this property is performed through the *MarkIntegrity* function as follows.

$$A[] \text{ MarkIntegrity}(T)$$

The *MarkIntegrity* function consists of two subfunction, namely *NoNotifyMark* and *AllMark*. *NoNotifyMark* assesses whether there is no *Notify* operation preceding a *Mark* operation. Simultaneously, *AllMark* determines whether all the answers have been appropriately marked. Through these checks, the integrity of the marking process is assured.

Validation and verification in UPPAAL

To validate the reliability of our electronic examination model, we conducted a verification process on the specified property specifications using UPPAAL. The verification experiment is performed on a computer equipped with an Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz 2.59 GHz, running Java 17.0.7, and utilizing UPPAAL version 5.0.0-rc3 as well as UPPAAL-5.1.0-beta5.

In our experiment, we focused on a scenario involving two candidates. It's noteworthy that more complex setups with additional candidates share similarities with this specific instance. The parameter configurations, detailed in Table 1, encompass a total of six parameters. N is set to 2, indicating two candidates. Q is set to 3, denoting three questions. M is set to 2, signifying two scores. $MaxSize$ is set to 50, indicating the maximum size of a queue is 50. $MaxT$ is set to 1000, denoting the maximum time (exam time) as 1000. Finally, the minimum permissible distance between candidates' answers is set to 1, denoted as D . As shown in Fig. 6, we simulate the electronic examination in this scenario using UPPAAL and validate the examination process with colleagues.

Related work

Enhancing the reliability and security of examinations is a crucial prerequisite to ensure accurate assessments of learners' knowledge and abilities. In the realm of electronic examinations, these properties become even more pivotal due to the incorporation of digital techniques. This overview summarizes existing research on the design of electronic examination process and the application of formal methods for electronic examination.

Design of electronic examination process

The design of electronic examination process encompasses the analysis of learner requirements and the construction of a robust system structure. Many researchers are dedicated to the meticulous process of requirement analysis and the implementation of frameworks for electronic examination systems.

Muzaffar et al. presented a systematic literature review of online examination, identified five leading features, discussed 16 important techniques/algorithms, 11 datasets as well as 21 online exam tools and investigated the participation of countries in online exam research(Muzaffar et al., 2021). Butler-Henderson et al. reviewed the contemporary literature on online examinations and explored 9 key themes, including student perceptions, student performance, anxiety, cheating, staff perceptions, authentication and security, interface, design and technology issues (Butler-Henderson and Crawford, 2020). Ilgaz et al. analyzed learners' academic achievement and perceptions in online exams at a public state university, showing that learners report positive

attitudes towards online exams and that there was no statistically significant difference in the students' academic achievement in online and traditional exams (Ilgaz and Afacan Adanır, 2020). Jiang et al. proposed a web-based online examination system using PHP, Ajax and other technologies, which has been applied to a course involving more than 1000 students per semester at Guangzhou University of Foreign Studies (Jiang et al., 2019).

Specifically, certain research endeavors concentrate on the various facets of cheating within electronic examination systems. Li et al. developed an optimization-based anti-collusion approach for distanced online testing (DOT) by minimizing the collusion gain, which can be coupled with other techniques for cheating prevention (Li et al., 2021). Ngqondi et al. used a literature review to understand academic fraud and respective security measures and propose a framework of online examinations for South African universities (Ngqondi et al., 2021). Noorbehbahani et al. presented a systematic review of research on cheating in online exams from 2010 to 2021, showed the categorization of the research and discussed topic trends in the field of online exam cheating (Noorbehbahani et al., 2022).

Analyzing requirements and implementing systems are crucial elements; however, the absence of essential quality assurance presents challenges for electronic examination systems that prioritize fairness. The application of formal methods proves beneficial in ensuring the reliability and security of the software process. The integration of formal methods into electronic examination systems significantly contributes to enhancing the overall system.

Formal methods for electronic examination

Formal methods cover the use of mathematically precise notations to specify and to reason about systems (Marmsoler, 2022). These methods, such as automata and Petri nets, prove valuable in enhancing the reliability and security of electronic examinations.

Kassem et al. proposed an event-based model of e-exams, defined several security properties and validated these properties by analyzing real e-exams at UGA using ProVerif and MarQ (Kassem et al., 2017). However, the explicit interactions between roles in electronic exams are not taken into consideration. In their models, two methods are employed, namely ProVerif and QEA (Quantified Event Automata). We consistently utilize the UPPAAL automaton to model all these processes.

Bella et al. proposed a secure exam protocol with the design principle of minimizing the reliance on the trusted parties, meeting a series of security requirements and resisting threats (Bella et al., 2017). However, this work primarily emphasizes cryptographic aspects, overlooking the absence of a visual and easily understandable modeling approach. A comprehensible model would significantly contribute to the explainability of electronic examinations, benefitting both system designers and teacher/student users.

Yang et al. proposed an approach for modeling online score system using hierarchical colored Petri nets and analyzed concurrency, conflict and causal dependency in CPN Tools (Xu et al., 2009). However, this work solely delves into an online score phase and its associated few properties. Our model comprehensively captures the entire processes of electronic examination and verifies a more extensive set of properties across all phases using UPPAAL.

In contrast to their work, our model is both executable and comprehensible. Leveraging the capabilities of UPPAAL, we present general templates and verify critical properties, thereby enhancing the trustworthiness of the electronic examination system.

Conclusions

This paper introduces a UPPAAL-based model for electronic examinations, focusing on both model specification and property specification. The model specification encompasses candidate, administrator, invigilator, and examiner templates. Property specification outlines 12 properties related to electronic examinations, covering aspects like candidate registration and exam availability. Utilizing UPPAAL, all properties are rigorously verified, and the results indicate that our model is reasonably trustworthy. This offers valuable guidance for system designers and teacher/student users alike.

In future work, we plan to integrate multiple administrators and invigilators into the model. Furthermore, we will broaden the scope of exam events to encompass other essential features and address more complex scenarios.

References

- Arcile, J. and André, É. (2022), "Timed automata as a formalism for expressing security: A survey on theory and practice", *ACM Computing Surveys*, Vol. 55 No. 6, pp. 127:1-127:36.
- Basile, D., Fantechi, A., Rucher, L. and Mandò, G. (2021), "Analysing an autonomous tramway positioning system with the UPPAAL Statistical Model Checker", *Formal Aspects of Computing*, Vol. 33 No. 6, pp. 957-987.
- Bella, G., Giustolisi, R., Lenzini, G. and Ryan, P. Y. A. (2017), "Trustworthy exams without trusted parties", *Computers & Security*, Vol. 67, pp. 291-307.
- Bobba, R., Grov, J., Gupta, I., Liu, S., Meseguer, J., Ölveczky, P. C. and Skeirik, S. (2018), "Survivability: Design, formal modeling, and validation of cloud storage systems using Maude", in Campbell, R. H., Kamhoua, C. A. and Kwiat, K. A. (Eds.) *Assured Cloud Computing*, Wiley-IEEE Press, pp. 10 - 48.
- Butler-Henderson, K. and Crawford, J. (2020), "A systematic review of online examinations: A pedagogical innovation for scalable authentication and integrity", *Computers & Education*, Vol. 159, pp. 104024:1-104024:12.
- Giustolisi, R. (2018), *Modelling and verification of secure exams*, Springer International Publishing.
- Hofmann, T. and Schupp, S. (2023), "Controlling timed automata against MTL specifications with TACoS", *Science of Computer Programming*, Vol. 225, pp. 102898:1-102898:5.
- Ilgaz, H. and Afacan Adanır, G. (2020), "Providing online exams for online learners: Does it really matter for them?", *Education and Information Technologies*, Vol. 25, pp. 1255-1269.

- Jiang, J., Wu, B., Chang, L., Liu, K. and Hao, T. (2019), "The design and application of an Web-based online examination system", *Proceedings of the 4th International Symposium on Emerging Technologies for Education*, pp. 246-256.
- Kassem, A., Falcone, Y. and Lafourcade, P. (2017), "Formal analysis and offline monitoring of electronic exams", *Formal Methods in System Design*, Vol. 51 No. 1, pp. 117-153.
- Lehmann, S. and Schupp, S. (2022), "Bounded DBM-based clock state construction for timed automata in Uppaal", *International Journal on Software Tools for Technology Transfer*, Vol. 25 No. 1, pp. 19-47.
- Li, M., Luo, L., Sikdar, S., Nizam, N. I., Gao, S., Shan, H., Kruger, M., Kruger, U., Mohamed, H., Xia, L. and Wang, G. (2021), "Optimized collusion prevention for online exams during social distancing", *npj Science of Learning*, Vol. 6, pp. 5:1-5:9.
- Marmsoler, D. (2022), "Review on modelling and verification of secure exams", *Formal Aspects of Computing*, Vol. 34 No. 2, pp. 1-3.
- Muzaffar, A. W., Tahir, M., Anwar, M. W., Chaudry, Q., Mir, S. R. and Rasheed, Y. (2021), "A systematic review of online exams solutions in E-Learning: Techniques, tools, and global adoption", *IEEE Access*, Vol. 9, pp. 32689-32712.
- Ngqondi, T., Maoneke, P. B. and Mauwa, H. (2021), "A secure online exams conceptual framework for South African universities", *Social Sciences & Humanities Open*, Vol. 3 No. 1, pp. 100132:1-100132:12.
- Noorbehbahani, F., Mohammadi, A. and Aminazadeh, M. (2022), "A systematic review of research on cheating in online exams from 2010 to 2021", *Education and Information Technologies*, Vol. 27 No. 6, pp. 8413-8460.
- Sakata, K., Fujita, S., Sawadab, K., Iwasawaa, H., Endoha, H. and Matsumoto, N. (2023), "Model verification of fallback control system under cyberattacks via UPPAAL", *Advanced Robotics*, Vol. 37 No. 3, pp. 156-168.
- UPPAAL Team (2023), "UPPAAL HELP", available at: <https://docs.uppaal.org/> (accessed 8 December 2023).
- Uppsala University and Aalborg University (2023), "UPPAAL Home", available at: <https://uppaal.org/> (accessed 8 December 2023).
- Xu, Y., Xie, X., Xia, D., Liu, Z. and Chen, L. (2009), "Modeling and analysis of an online score system using Colored Petri Nets", *Proceedings of the 3rd International Conference on Anti-counterfeiting, Security, and Identification in Communication*, pp. 1-5.

Figure 1

Interactions among examination participants.

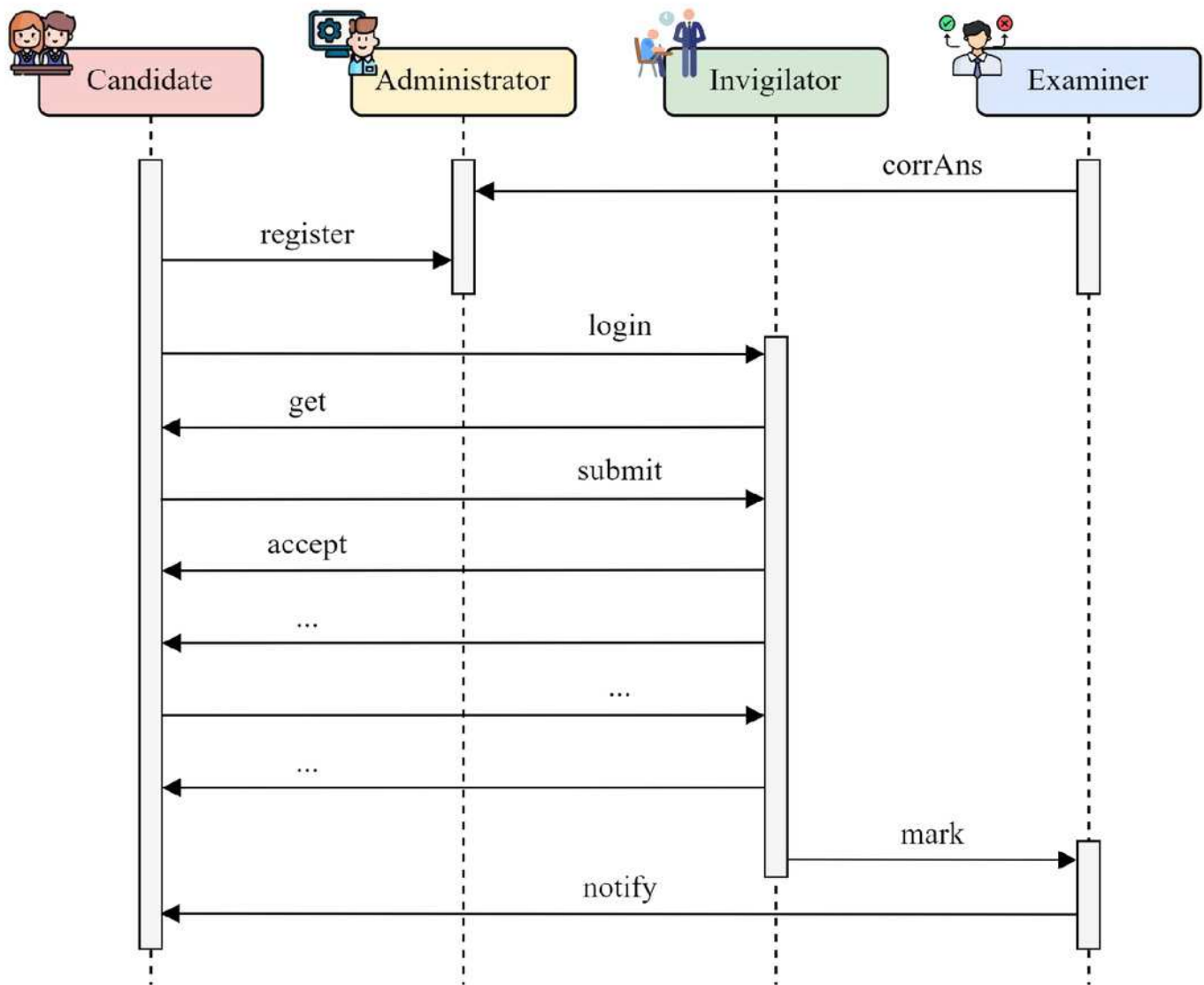


Figure 2

Candidate template.

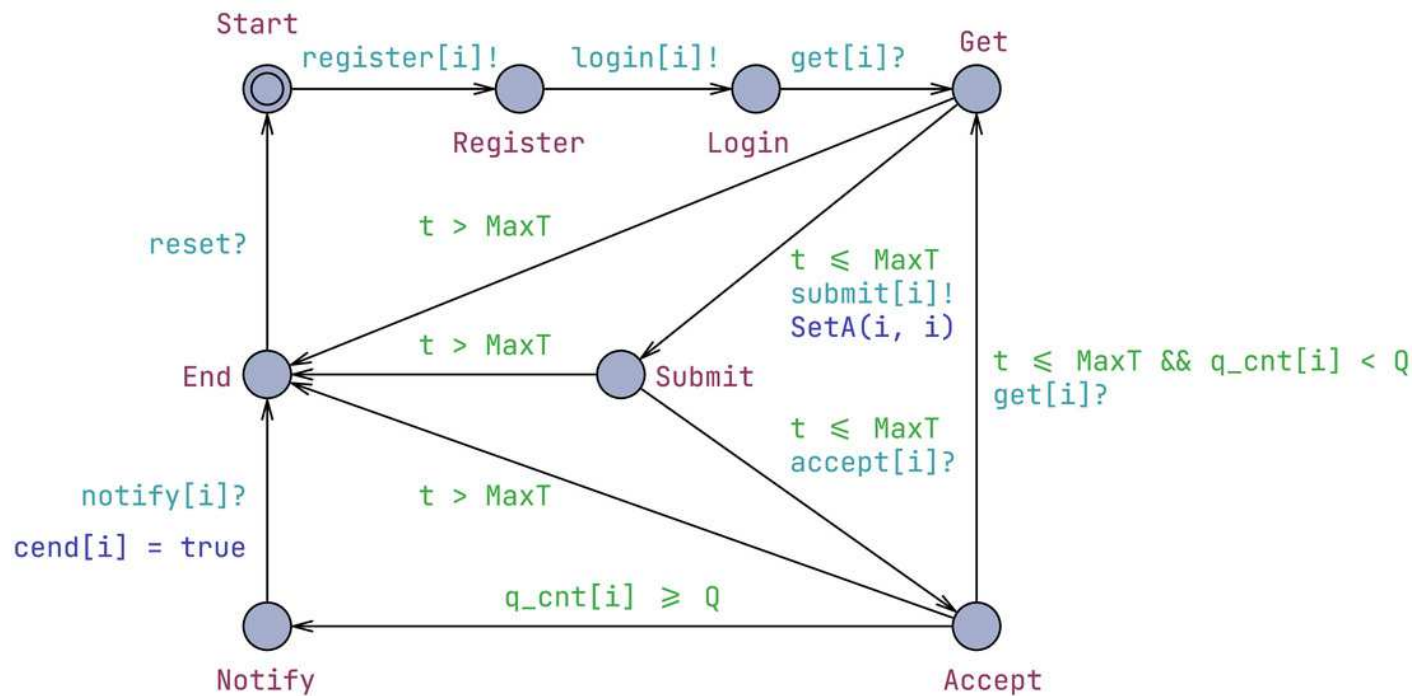


Figure 3

Administrator template.

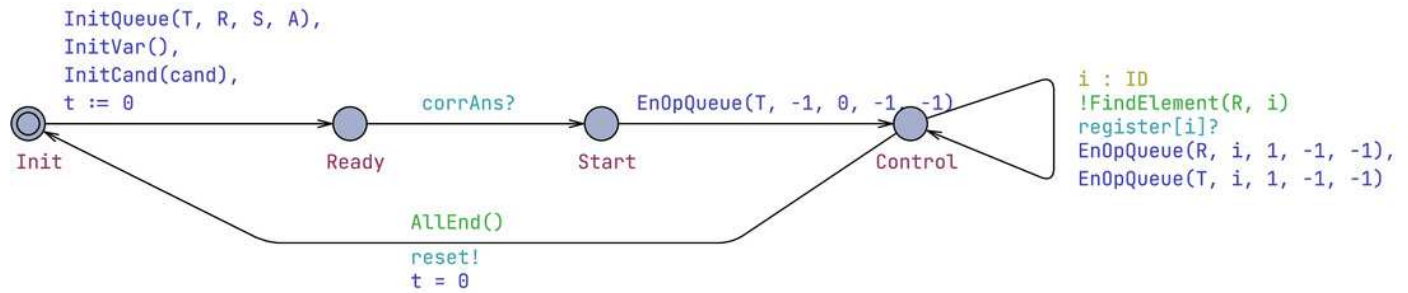


Figure 4

Invigilator template.

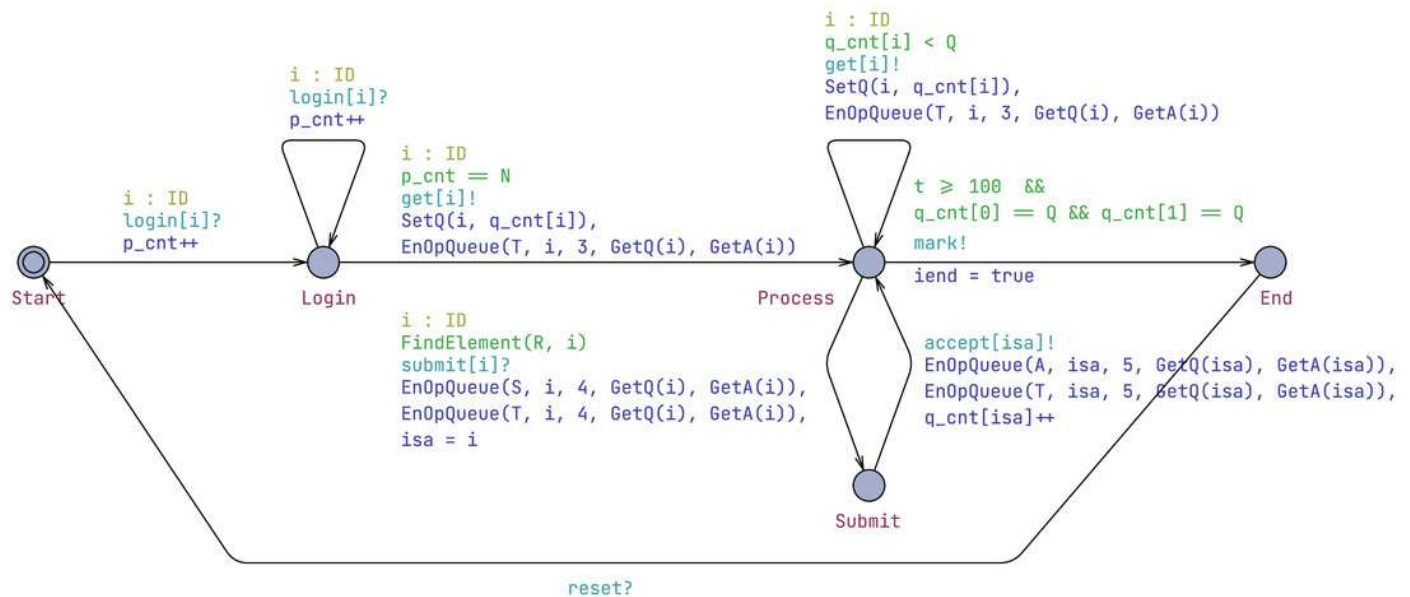
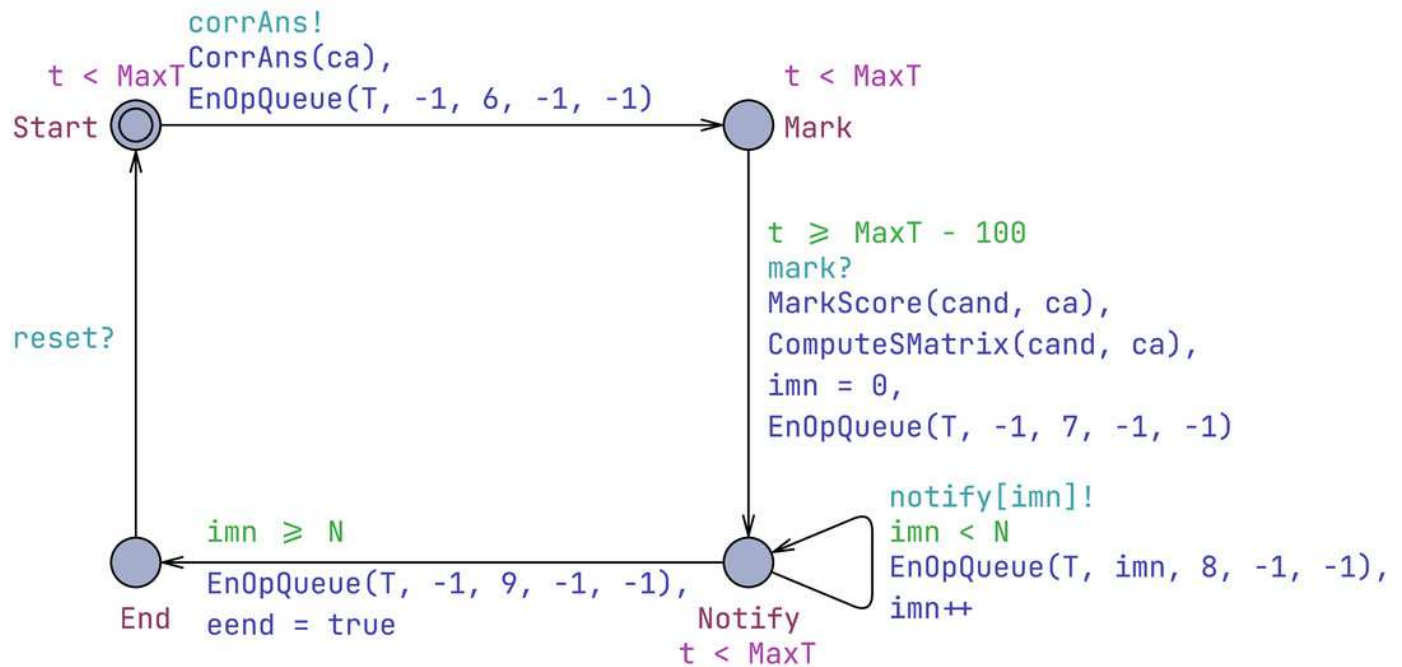


Figure 5

Examiner template.



Simulation and validation.

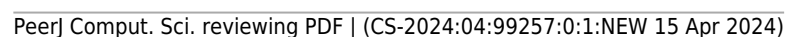


Figure 7

Verification results.

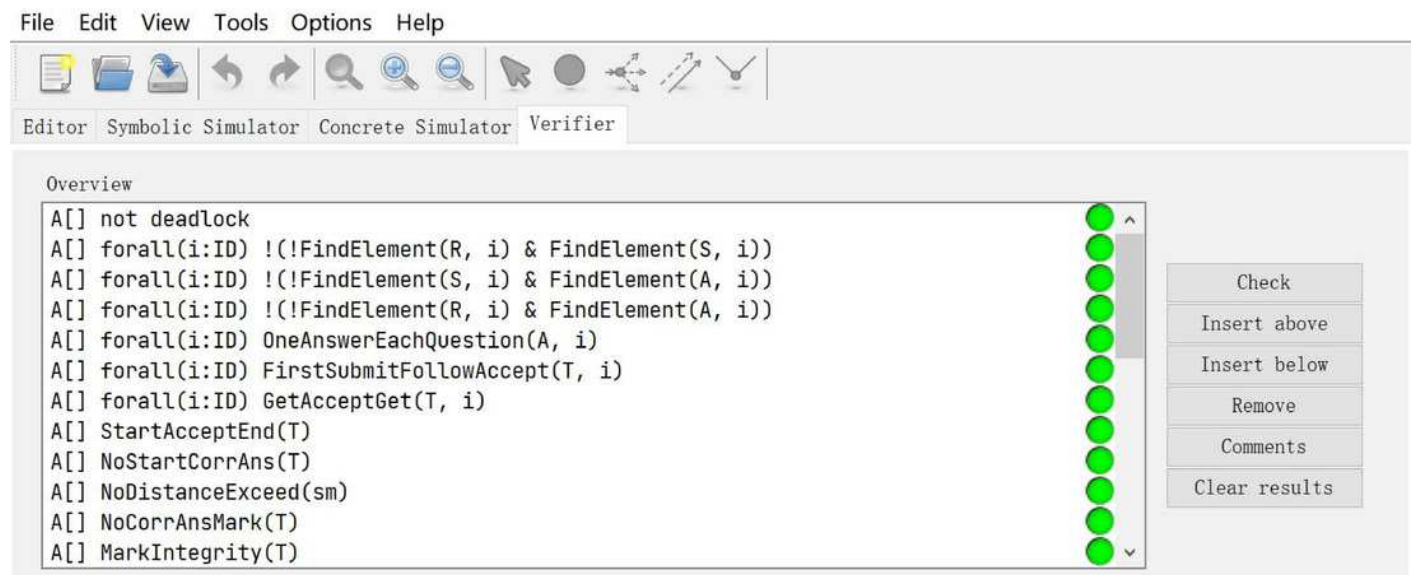


Table 1 (on next page)

Parameter settings.

1 Table 1:
 2 Parameter settings.

| Parameter | Value | Meaning |
|-----------|-------|--|
| N | 2 | the number of total candidates |
| Q | 3 | the number of total questions |
| M | 2 | the number of total scores |
| MaxSize | 50 | the max size of a queue |
| MaxT | 1000 | the max time (exam time) |
| D | 1 | the minimum permissible distance between candidates' answers |

3

Table 2 (on next page)

Verification results.

1 Table 2:
2 Verification results.

| No. | Property | Verification time/Kernel time/Total time (s) | Resident memory/Virtual memory peak (KB) | Result |
|------|------------------------|--|---|-----------|
| (1) | No deadlock | 0.922s / 0s / 0.926s | 25, 492KB / 78, 064KB | Satisfied |
| (2) | Candidate registration | 0.718s / 0.016s / 0.734s | 28, 356KB / 83, 724KB | Satisfied |
| (3) | Candidate eligibility | 0.734 s / 0.016s / 0.747s | 30, 932KB / 8, 960KB | Satisfied |
| (4) | Answer authentication | 0.719s / 0s / 0.722s | 33, 572KB / 94, 440KB | Satisfied |
| (5) | Answer singularity | 1.156 s / 0.015s / 1.178s | 36, 380KB / 99, 964KB | Satisfied |
| (6) | Acceptance assurance | 1.687s / 0s / 1.698s | 38, 740KB / 104, 676KB | Satisfied |
| (7) | Questions ordering | 1.063s / 0s / 1.053s | 41, 732KB / 110, 720KB | Satisfied |
| (8) | Exam availability | 0.766s / 0s / 0.763s | 44, 112KB / 115, 564KB | Satisfied |
| (9) | Answer-score integrity | 0.625s / 0.032s / 0.653s | 46, 972KB / 121, 236KB | Satisfied |
| (10) | Cheater detection | 0.5s / 0.015s / 0.516s | 49, 924KB / 127, 280KB | Satisfied |
| (11) | Marking correctness | 0.625s / 0s / 0.642s | 52, 604KB / 132, 664KB | Satisfied |
| (12) | Mark integrity | 0.687s / 0s / 0.691s | 55, 260KB / 138, 008KB | Satisfied |

3