Modeling and Specification of EasyPoltrona

-

A Scientific Conference Management System

André Gabriel Butuc
Bruno Emanuel Prata de Matos
José Pedro Santos Mendes
Rui Pedro Rosas Soares

FEUP/MESW/PPES, December 2024

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Table of Contents

1. Introduction

1.1. Objectives

This document contains semi-formal models in UML and formal specifications in OCL of **EasyPoltrona, a Scientific Conference Management System (SCMS)**.
It aims to define and analyze business processes, use cases, architecture, and domain entities to support functionalities like submission, review, scheduling, and participant registration.

1.2. System overview

We assume the EasyPoltrona Scientific Conference Management System supports the following requirements:

R1. Users must create an account on EasyPoltrona to access its features.

R2. The system must send email notifications for verification outcomes and other relevant events.

R3. Users can create a conference by filling out a form with the conference's details.

R4. The system must verify the conference details within a maximum of one day.

R5. Conference chairs are automatically assigned as the first chair of the conference after approval.

R6. Approved conferences can accept paper submissions.

R7. Users can register to attend a conference by selecting their type (Author, Regular, or Student) and food preferences.

R8. The system must calculate and display the appropriate registration fee based on the participant type.

R9. The system must support tracking of payment status for registrations.

R10. The system must send periodic reminders for unpaid invoices.

R11. Conference chairs can publish conference proceedings by integrating with external publishing services.

R12. Only reviewed and accepted papers can be published.

R13. Authors can submit papers via the system by selecting a track and filling out a submission form with paper details.

R14. Authors can upload the paper file and provide metadata such as title, abstract, keywords, and authors.

R15. Conference chairs can assign reviewers to papers for review.

R16. The system must notify reviewers when they are assigned to review a paper.

R17. Reviewers can provide reviews that include overall evaluation (7 possible grades), reviewer confidence (5 possible levels), review text, and confidential remarks.

R18. Confidential remarks are visible only to conference chairs.

R19. Conference chairs can make final decisions to accept or reject papers based on gathered reviews.

R20. Conference chairs can send review feedback to authors for enhancing and improving their papers.

R21. Conference chairs can create an open call for submissions by filling out a form with conference details and submission deadlines.

R22. Users can search for conferences using criteria like research area, country, topics, and tags.

R23. The system must send reminders to authors about submission deadlines.

R24. Authors can upload presentation slides via the system to share research results.

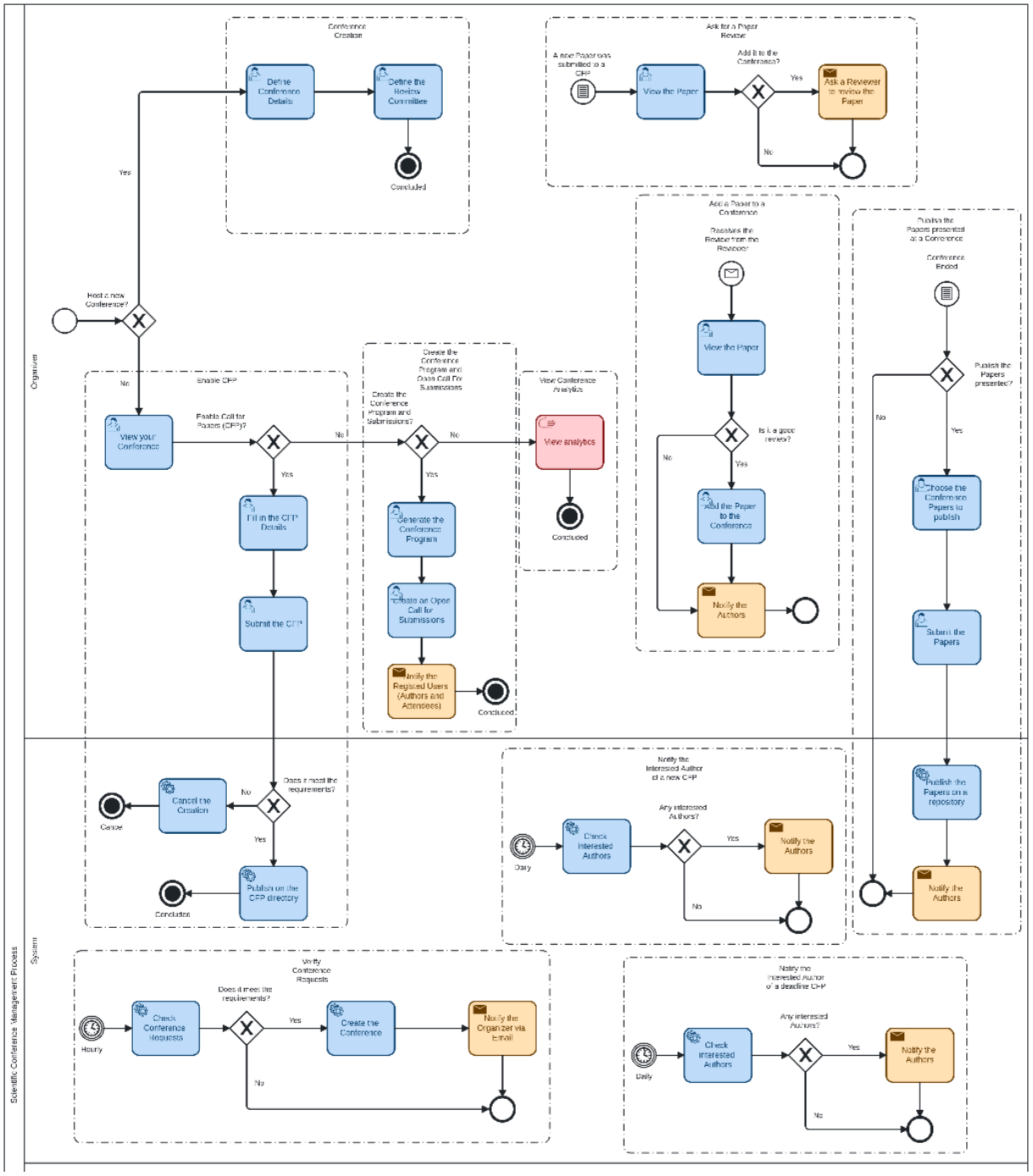
R25. Conference chairs can generate a detailed conference program by specifying talk schedules, presentations, and breaks.

1.3. Document organization

Next sections 2, 3, 4, and 5 present different semi-formal models (or views) of the system, namely, a business process model, a use case model, a domain model, and an architectural model, based on different types of UML diagrams [1], drawn with the Enterprise Architect tool [3], plus associated documentation in natural language. Section 6 presents a formalization and validation of the domain model using the OCL [2] language and the USE [4] tool.

2. Business process model

2.1. Overview



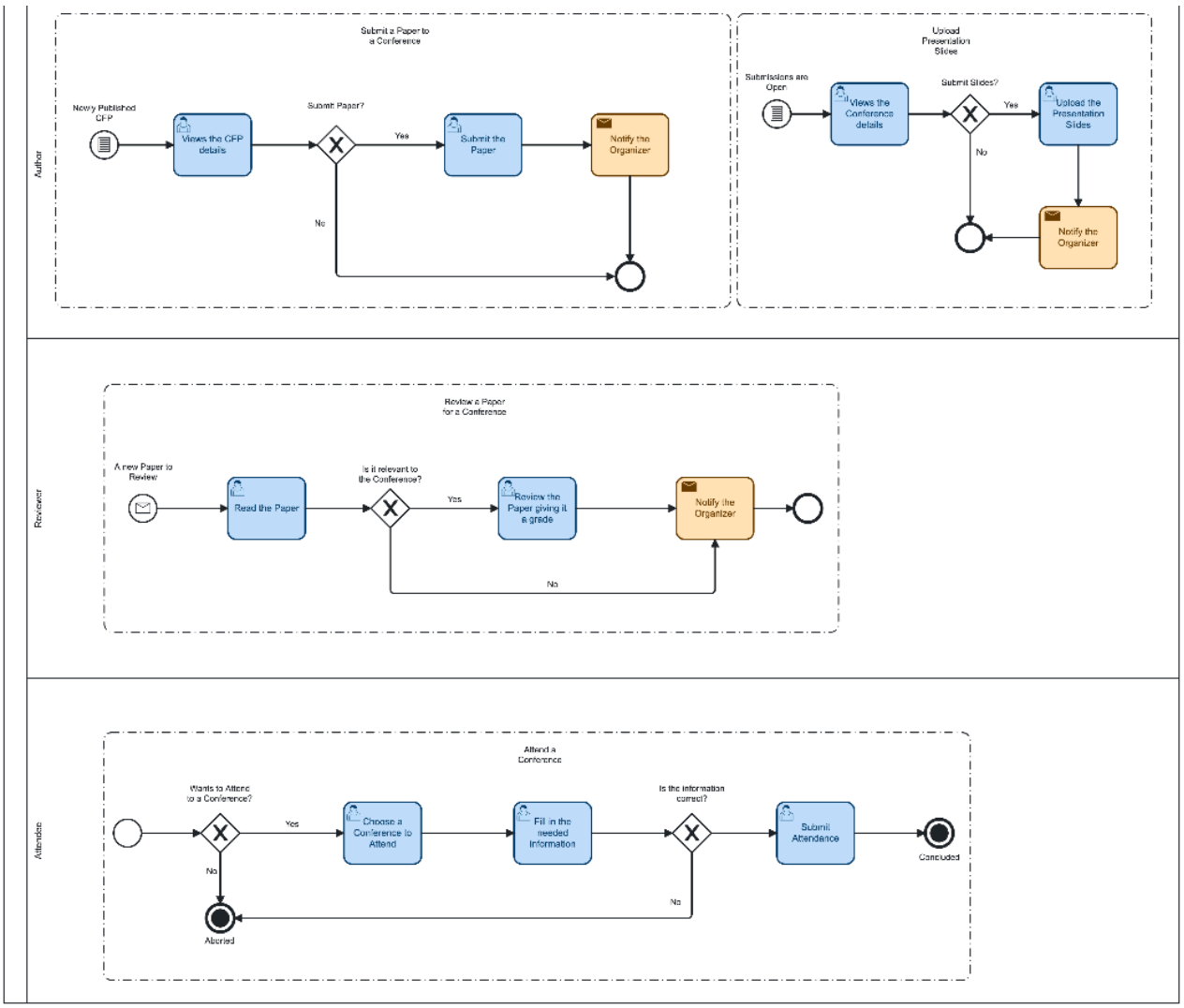


Figure 1. SCMS diagram describing the setup of a conference and subprocesses (created with Camunda).

2.2. Subprocess descriptions

Subprocess	Description
Conference Creation	Describes the process where a user creates a conference by filling out a form with the conference's details and assigning the review committee that will review the submitted papers. The system verifies these details within a maximum of one day.
Ask for a Paper Review	Describes the subprocess where conference chairs assign reviewers to papers, notify them, and track the review submission.
Enable CFP	Describes the process where conference chairs create and publish a Call for Papers (CFP) by providing details such as conference topics, tracks, and submission deadlines.
Create the Conference Program and Open Call For Submissions	Describes the process where conference chairs generate the detailed conference program, including talk schedules and breaks, and open submissions for the users (Attendees and Authors).
View Conference Analytics	Describes the process where users, particularly conference chairs, can view statistics and insights about the conference, such as the number of submissions, registrations, or payment statuses.
Add a Paper to a Conference	Describes the subprocess where authors submit their papers by selecting a track, providing metadata, and uploading the paper file.
Publish the Papers presented at a Conference	Describes the process where, after review and acceptance, conference chairs integrate with external publishing services to publish the conference proceedings.
Notify the Interested Author of a new CFP	Daily system process that sends automated notifications to registered authors based on their research interests, informing them of new CFPs for upcoming conferences.
Verify Conference Requests	Hourly system process that validates conference details submitted by users to ensure accuracy and completeness before approval.
Notify the Interested Author of a deadline CFP	Daily system process where the Authors are reminded via email notifications about approaching deadlines for CFP submissions to ensure timely submissions.
Submit a Paper to a Conference	Describes the process where Authors submit their research papers by providing details like title, abstract, keywords, and co-authors, and selecting the relevant track.
Upload Presentation Slides	Describes the process where Authors upload their presentation slides to the system before the conference to share research results during the event.
Review a Paper for a Conference	Describes the process where Reviewers evaluate assigned papers by providing a detailed review, including grades, confidence levels, review text, and confidential remarks for conference chairs.

3. Use case model

Section 3.1 presents an overview of the system actors and use cases, derived directly from the business processes model in Section 2 and the requirements in Section 1.2. Section 3.2 presents brief actor descriptions. Section 3.3 presents detailed use case specifications.

3.1. Overview - use case diagram

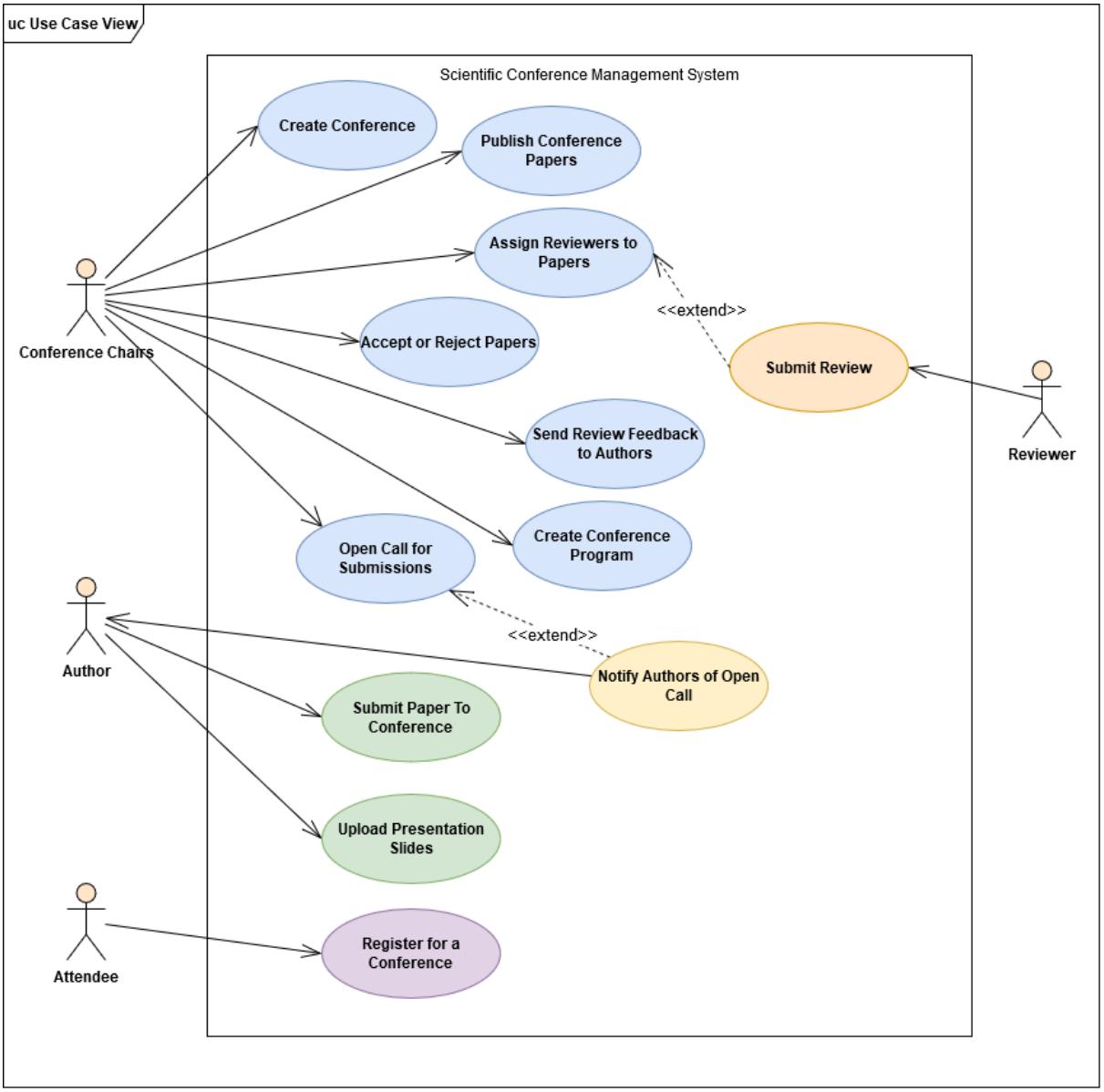


Figure 2. UML use case diagram showing the main actors and use cases in the system.

The Conference Chairs are responsible for:

- creating conferences;
- publish conference papers;
- assign reviewers to papers;
- accept or reject papers;
- send review feedback to authors;

- create conference program;
- open call for submissions, notifying authors of such open calls.

Authors may:
- submit paper to conferences;
- upload presentation slides.

Attendees may:
- register for a conference.

Reviewers are responsible for:
- Submitting reviews.

The system is responsible for sending automatic notifications to Authors when:
- Conference Chairs open call for submissions

3.2. Actor descriptions

Actor	**Description**
Conference Chair	The individual responsible for organizing and managing a conference on EasyPoltrona. They oversee key activities such as conference creation, paper submission management, assigning reviewers, making acceptance decisions, and generating the conference program.
Author	A researcher or academic who submits their papers to conferences hosted on EasyPoltrona. Authors can also upload presentation slides and receive review feedback to enhance their work.
Reviewer	A subject-matter expert assigned by the Conference Chair to evaluate and review submitted papers. Reviewers provide detailed assessments, including grades, confidence levels, and feedback to help determine the paper's acceptance.
Attendee	An individual who registers to attend a conference. Attendees can be categorized as Authors, Regular participants, or Students, each of whom may have specific registration requirements and fees.

3.3. Use case specifications

Use case	**Create Conference**
Description	Allows users to create a conference by submitting details for verification.
Preconditions	● The user must have an EasyPoltrona account.
Post-conditions	● The conference is either approved or denied and listed accordingly on the platform. ● An email notification about the verification outcome is sent to the user.
Flow of events	1. The Conference Chair logs into EasyPoltrona. 2. The Chair completes a form with conference details. 3. The system submits the conference for verification. 4. The system verifies the conference within one day. 5. The system sends an email to the Chair with the verification outcome. 6. If approved, the conference is listed on the platform and ready for submissions.
Exceptions	a) If the verification fails due to incomplete or invalid information the system rejects the request and notifies the user to resubmit with corrections.

Use case	**Register for a Conference**
Description	Allows users to register as attendees for a conference.
Preconditions	● The conference must be approved and available for registration.
Post-conditions	● The attendee registration is recorded. ● Payment status is tracked by the system.
Flow of events	1. The Attendee logs into EasyPoltrona. 2. The Attendee selects the conference to attend. 3. The Attendee fills out a form, specifying attendee type (Author, Regular, Student) and food preferences. 4. The system calculates and displays the registration fee. 5. The Attendee confirms the registration. 6. The system tracks the payment status and sends reminders for unpaid invoices as needed.
Exceptions	a) If the registration fee is not paid within the deadline, the registration is canceled, and the system notifies the attendee.

Use case	**Publish Conference Papers**
Description	Enables conference chairs to publish accepted papers as conference proceedings.
Preconditions	● Paper submissions and reviews must be completed for the conference.
Post-conditions	● The accepted papers are published as proceedings through an external service.
Flow of events	1. The Conference Chair logs into EasyPoltrona. 2. The Chair selects the target conference. 3. The Chair initiates integration with an external publishing service. 4. The system publishes the accepted papers as proceedings.
Exceptions	a) If integration with the external service fails, the system notifies the Conference Chair and logs the issue for resolution.

Use case	**Submit Paper to Conference**
Description	Allows authors to submit papers for a conference.
Preconditions	● The conference must be open for submissions.
Post-conditions	● The paper submission is recorded and linked to the selected track.
Flow of events	1. The Author logs into EasyPoltrona. 2. The Author selects the target conference. 3. The Author chooses a track for the paper. 4. The Author fills out the submission form, providing details like title, abstract, keywords, and authors. 5. The Author uploads the paper file. 6. The system confirms the submission.
Exceptions	a) If the paper upload fails, the system notifies the Author to retry. b) If the submission deadline has passed, the system prevents submission and notifies the Author.

Use case	**Assign Reviewers to Papers**
Description	Enables conference chairs to assign reviewers for submitted papers.
Preconditions	● Papers must be submitted to the conference.
Post-conditions	● Reviewers are notified of their assignments. ● Reviews are recorded in the system.
Flow of events	1. The Conference Chair logs into EasyPoltrona. 2. The Chair selects a paper and assigns a reviewer. 3. The system notifies the assigned reviewer. 4. The Reviewer logs into EasyPoltrona and submits the review, including evaluation, confidence, review text, and confidential remarks. 5. The system records the review.
Exceptions	a) If a reviewer declines the assignment, the system notifies the Chair to reassign the paper.

Use case	**Accept or Reject Papers**
Description	Enables conference chairs to accept or reject paper submissions based on reviews.
Preconditions	● Reviews must be completed for the paper.
Post-conditions	● The paper status is updated to "Accepted" or "Rejected".
Flow of events	1. The Conference Chair logs into EasyPoltrona. 2. The Chair reviews the gathered reviews for a paper. 3. The Chair makes a decision to accept or reject the paper. 4. The system updates the paper's status accordingly.
Exceptions	a) If reviews are missing or incomplete, the system alerts the Chair and prevents the decision until all reviews are submitted.

Use case	**Send Review Feedback to Authors**
Description	Enables conference chairs to send reviews and feedback to authors.
Preconditions	● Paper reviews must be completed.
Post-conditions	● The authors receive feedback for their paper submissions.
Flow of events	1. The Conference Chair logs into EasyPoltrona. 2. The Chair selects a paper and its associated reviews. 3. The Chair sends the review feedback to the Author. 4. The system delivers the feedback to the Author.
Exceptions	a) If the feedback cannot be delivered (e.g., due to an invalid email address), the system logs the issue and notifies the Chair.

Use case	**Open Call for Submissions**
Description	Allows conference chairs to create and advertise a call for submissions.
Preconditions	● The conference must be approved.
Post-conditions	● The call for submissions is published and searchable by authors. ● Submission deadline reminders are sent.
Flow of events	1. The Conference Chair logs into EasyPoltrona. 2. The Chair completes a form with conference details, including submission deadlines. 3. The system publishes the call for submissions. 4. Authors search for conferences using criteria like topics and tags. 5. The system sends reminders to authors about submission deadlines.
Exceptions	a) If the call for submissions is incomplete, the system notifies the Chair and prevents publication.

Use case	**Upload Presentation Slides**
Description	Allows authors to upload presentation slides for dissemination.
Preconditions	● The conference must accept presentations.
Post-conditions	● The presentation slides are stored and linked to the author's paper or presentation.
Flow of events	1. The Author logs into EasyPoltrona. 2. The Author selects the conference. 3. The Author uploads the presentation slides. 4. The system confirms and stores the uploaded slides.
Exceptions	a) If the upload fails, the system notifies the Author to retry.

Use case	**Create Conference Program**
Description	Enables conference chairs to create a detailed program for the conference.
Preconditions	● The conference must be approved.
Post-conditions	● The conference program is published and available to participants.
Flow of events	1. The Conference Chair logs into EasyPoltrona. 2. The Chair specifies details for talks, presentations, and breaks. 3. The system generates the conference program. 4. The system publishes the program.
Exceptions	a) If the program is incomplete, the system prevents publication and notifies the Chair.

4. Domain model

4.1. Overview - class diagram

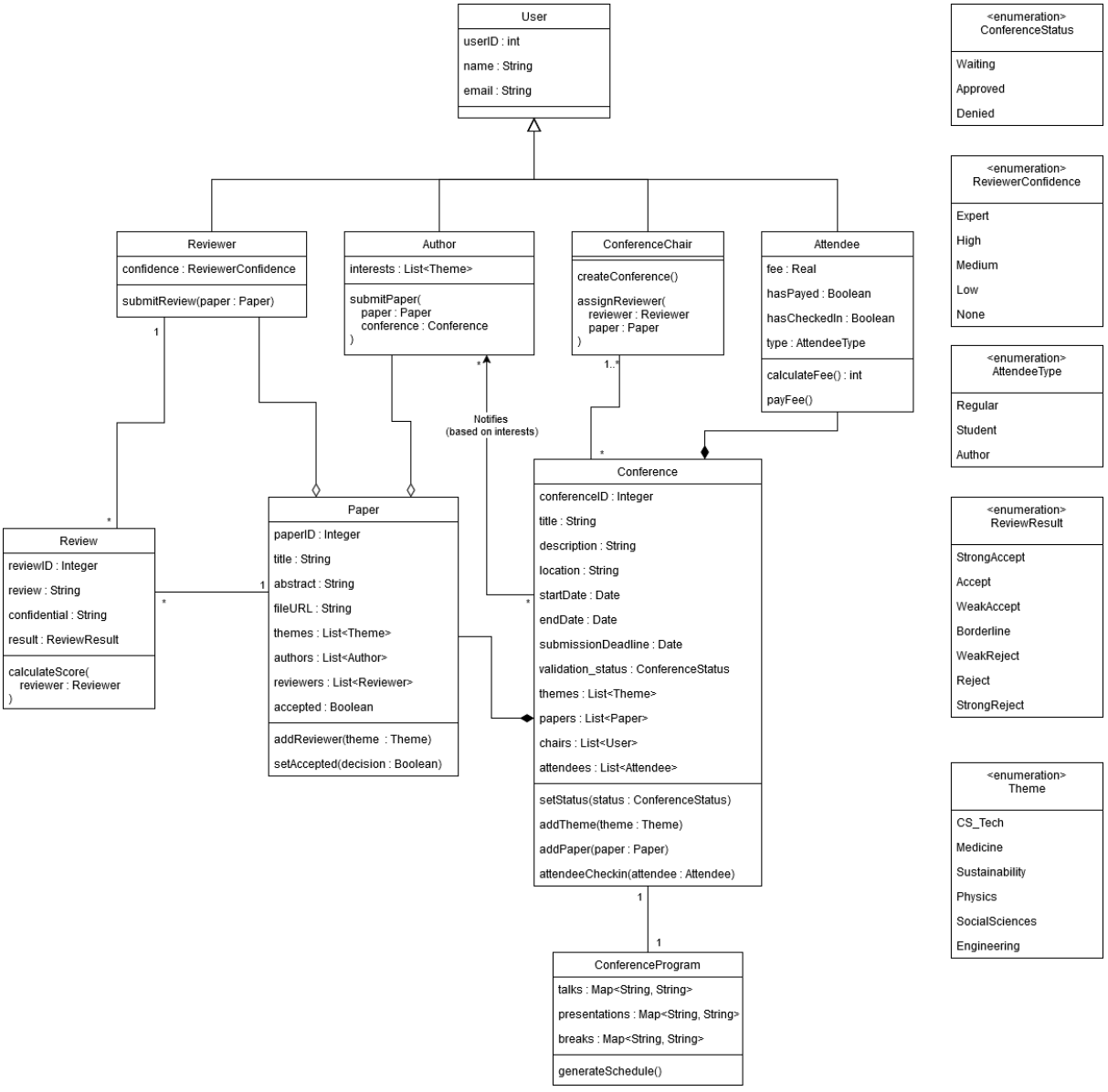


Figure 3. UML class diagram showing the main entities, relationships, attributes, and transactions in the system.

Entity or datatype	Description
User	Represents a user with basic details like ID, name, and email.
Reviewer	A user who reviews papers, with an associated confidence level.
Author	A user who submits papers and has specific areas of interest (themes).
Conference Chair	A user who manages the conference, creates it, and assigns reviewers to papers.

Conference	Represents a conference with details such as title, location, themes, papers, and attendees.
Paper	A research paper with details like title, abstract, file URL, associated themes, authors, and status.
Review	A review for a paper, containing the review text, confidential remarks, and a result status.
Attendee	Represents a person attending the conference with attributes like fee, payment status, and type.
ConferenceProgram	Details the schedule of talks, presentations, and breaks for the conference.
ConferenceStatus	Enumeration for the validation status of a conference (Waiting, Approved, Denied).
ReviewerConfidence	Enumeration for the confidence level of a reviewer (Expert, High, Medium, Low, None).
ReviewResult	Enumeration for the outcome of a review (e.g., StrongAccept, Reject).
AttendeeType	Enumeration for the type of an attendee (Regular, Student, Author).
Theme	Enumeration of themes/topics for papers and conferences (e.g., CS_Tech, Medicine).

Table 2. Brief description of the main entities and data types in the system.

4.2. Integrity constraints

On User:

I01. The userID must be unique.

I02. The email must be unique and in a valid email format .

I03. The name cannot be null or empty.

On Reviewer:

I04. Confidence must be one of the valid ReviewerConfidence values.

I05. A Reviewer can only submit reviews for assigned papers.

On Author:

I06. Interests must contain valid Theme values.

I07. An Author can submit only papers they authored.

On Attendee:

I08. The type must be one of the valid AttendeeType values.

I09. The fee must be a non-negative real number.

I11. hasCheckedIn can only be true if hasPayed is also true.

On Conference:

I11. ConferenceID must be unique.

I12. Title, description and location cannot be null or empty.

I13. The endDate must be after the startDate.

I14. The startDate must be after submissionDeadline.

I15. At least one Paper must be listed in papers.

I16. At least one Theme must be listed in themes.

I17. At least one User must be listed in chairs.

On Paper:

I18. The paperID must be unique.

I19. The title, abstract and fileURL cannot be null or empty.

I20. At least one Theme must be listed in themes.

I21. At least one Author must be listed in authors.

On Review:

I22. reviewID must be unique.

I23. result must be one of the valid ReviewResult values.

On ConferenceProgram:

I24. Keys in talks, presentations, and breaks must be unique.

I25. All times in the program must be within the startDate and endDate of the Conference.

4.3. Operation specifications

Name and Syntax	**Reviewer::submitReview**(review: Review, paper: Paper)
Description	Allows a reviewer to submit a review for a specified paper.
Preconditions	● The Reviewer must be assigned to the Paper. ● The Paper must exist.
Post-conditions	● The Review is added to the Paper's list of reviews.

Name and Syntax	**Author::submitPaper**(paper: Paper, conference: Conference)
Description	Allows an author to submit a paper to a specific conference.
Preconditions	● The Conference must exist and have a status of Approved. ● The Paper must include valid information such as title, abstract, fileURL, and at least one Author.
Post-conditions	● The Paper is added to the Conference's list of papers.

Name and Syntax	**ConferenceChair::createConference**(): Conference
Description	Allows the conference chair to create a new conference with initial attributes.
Preconditions	● The chair must have the necessary privileges.
Post-conditions	● A new Conference object is created with default attributes. ● The Conference is in Waiting status.

Name and Syntax	**ConferenceChair::assignReviewer**(reviewer: Reviewer, paper: Paper)
Description	Assigns a reviewer to a specific paper.
Preconditions	● The Reviewer must exist and have a valid ReviewerConfidence level. ● The Paper must belong to the Conference. ● The Reviewer should have expertise in at least one of the Paper's themes.
Post-conditions	● The Reviewer is added to the Paper's list of reviewers.

Name and Syntax	**Conference::setStatus**(status: ConferenceStatus)
Description	Updates the conference's validation status.
Preconditions	● The status must be one of the valid ConferenceStatus values. ● The current status must allow a change (e.g., cannot change from Denied to Approved).
Post-conditions	● The validation_status of the Conference is updated.

Name and Syntax	**Conference::addTheme**(theme: Theme)
Description	Adds a theme to the conference.
Preconditions	● The theme must be one of the valid Theme values.
Post-conditions	● The Theme is added to the Conference's list of themes.

Name and Syntax	**Conference::addPaper**(paper: Paper)
Description	Adds a submitted paper to the conference.
Preconditions	● The Paper must be valid and meet all submission criteria (e.g., format, deadline).
Post-conditions	● The Paper is added to the Conference's list of papers.

Name and Syntax	**Conference::attendeeCheckIn**(attendee: Attendee)
Description	Marks an attendee as checked in for the conference.
Preconditions	● The Attendee must be registered for the Conference. ● The Attendee must have paid their registration fee.
Post-conditions	● The hasCheckedIn status for the Attendee is set to true.

Name and Syntax	**Paper::addReviewer**(reviewer: Reviewer)
Description	Adds a reviewer to the paper based on their expertise in the specified theme.
Preconditions	● The Reviewer must have expertise in the Theme. ● The Reviewer must not already be assigned to the Paper.
Post-conditions	● The Reviewer is added to the paper's list of reviewers.

Name and Syntax	**Paper::addTheme**(theme: Theme)
Description	Adds a new theme to the paper based.
Preconditions	● The Theme must not already be one of the existing themes in the Paper.
Post-conditions	● The Theme is added to the paper's list of themes.

Name and Syntax	**Paper::acceptPaper**()
Description	Updates the acceptance status of a paper.
Preconditions	● There must be at least 3 positive submitted reviews of the paper.
Post-conditions	● The accepted attribute of the Paper is updated to true.

Name and Syntax	**Attendee::calculateFee**(): Real
Description	Calculates the registration fee for the attendee based on their type (Regular, Student, Author).
Preconditions	● The type of the Attendee must be set.
Post-conditions	● The calculated fee is returned.

Name and Syntax	**Attendee::payFee**(): void
Description	Marks the attendee's fee as paid.
Preconditions	● The calculated fee for the Attendee must be greater than or equal to 0.
Post-conditions	● The hasPayed status of the Attendee is set to true.

Name and Syntax	**ConferenceProgram::generateSchedule**(): void
Description	Automatically generates a schedule for the conference, including talks, presentations, and breaks.
Preconditions	● The Conference must have at least one paper and associated talks/presentations. ● The startDate and endDate must be defined.
Post-conditions	● A schedule is generated and populated in the ConferenceProgram.

4.4. Object lifecycles

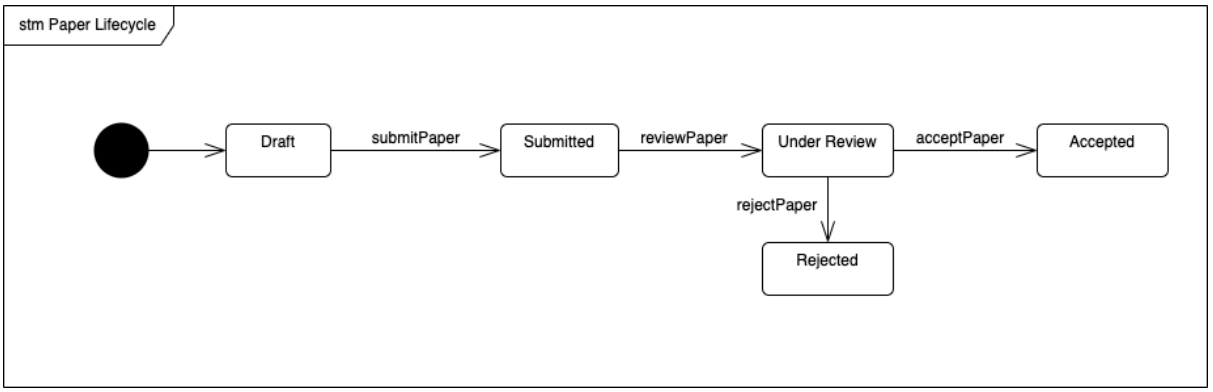


Figure 4. UML state machine diagram showing the lifecycle of a Paper.

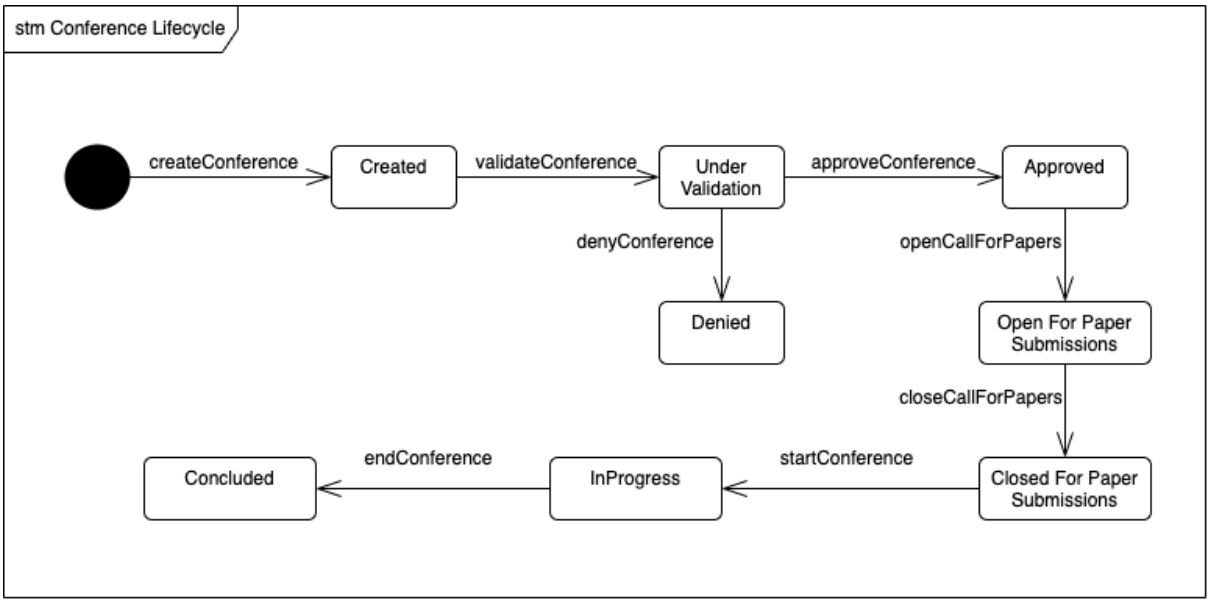


Figure 5. UML state machine diagram showing the lifecycle of a Conference.

5. Architectural model

The system follows a classic client-server architecture. A deployment view of the system architecture is shown in Section 5.1 using a UML deployment diagram. The involved hardware nodes and the software and data artifacts deployed in those nodes are described in sections 5.2 and 5.3, respectively.

5.1. Overview - deployment diagram

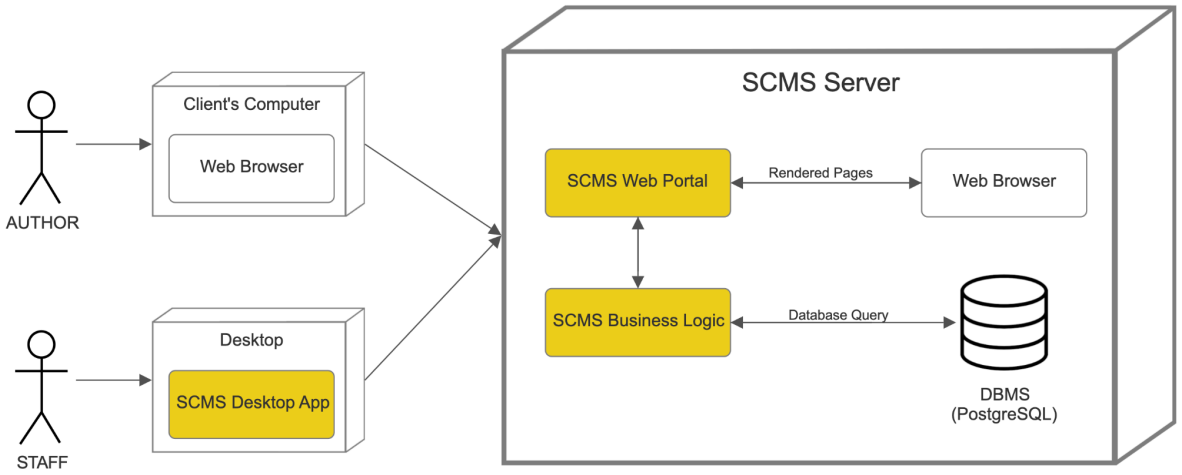


Figure 6. UML deployment diagram for the SCMS system.

5.2. Hardware nodes

Hardware Node	Description
SCMS Server	A server-side computer (running Linux or another server-side OS) hosting SCMS backend components, database, and web server.
Author' Computer	A device (e.g., desktop or laptop) used by authors to access the SCMS Web Portal for submitting and managing papers.
STAFF Computer	A workstation used by organizers to manage conference schedules, reviews, and settings via the SCMS Web Portal.

5.3. Software and data artifacts

Software/Data Artifact	Description
SCMS Web Portal	A web-based interface providing functionality for authors, reviewers, attendees, and organizers.
SCMS Business Logic	Core system logic, implementing functionalities like submission management, review assignments, and scheduling.
SCMS Database	A relational database storing system data such as submissions, reviews, user details, and schedules.

SCMS Desktop App	A desktop application installed on the Organizer's Workstation, providing administrative functionalities such as managing submissions, reviews, and conference settings.
Web Browser	A software used on client devices to access the SCMS Web Portal.

6. Domain model formalization and validation

6.1. Domain model formalization

An encoding of the domain model defined in Section 4 (UML class diagram elements, class invariants and operations pre/post-conditions) in OCL and USE is presented in Appendix A.

A corresponding class diagram obtained with the USE tool is shown in Figure 7.

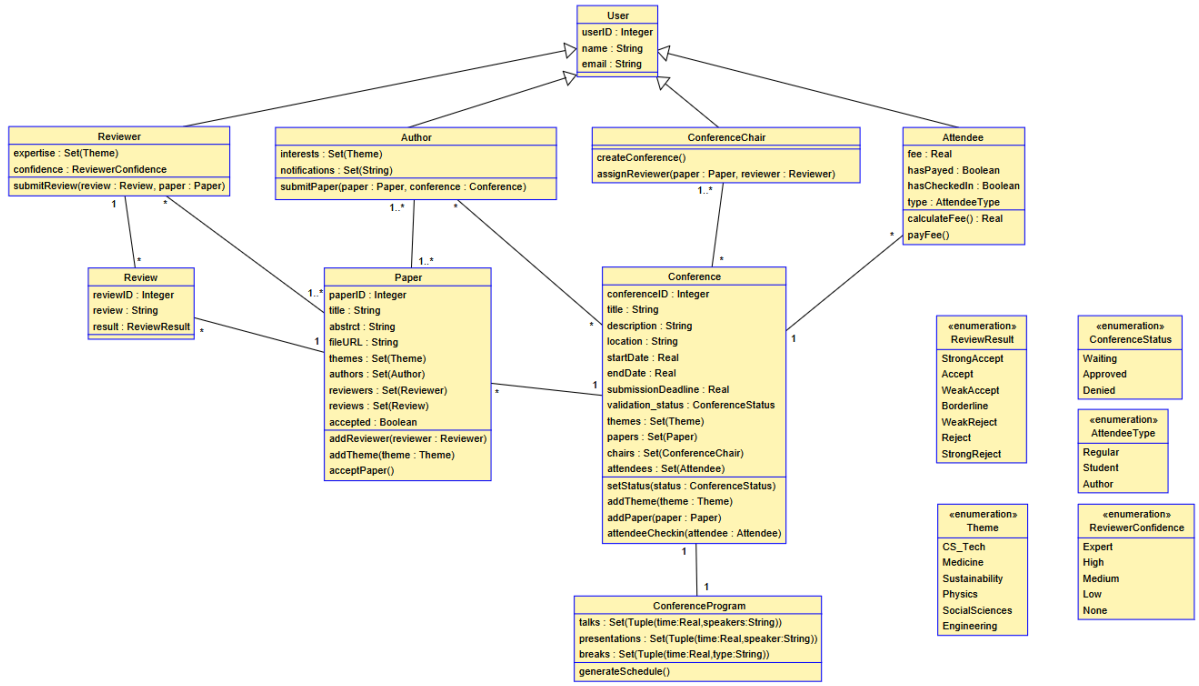


Figure 7. Class diagram generated with the USE tool from the given specification.

In this encoding, some adaptations have been made:

- to facilitate operating with date and time values, they are represented by real numbers, representing the time elapsed since a reference time instant;
- to simulate the sending of notifications, a new field `notifications` was added to the class `Author` to store the notifications sent (as a set of strings);
- to facilitate debugging, all elementary pre/post-conditions are defined in separate `pre` and `post` clauses.

6.2. Domain model validation

A test scenario described in the USE command language is presented in Appendix B.

The objects involved in this scenario, with the corresponding state at the end of the test scenario, are shown in the object diagram of Figure 8. For some reason we could not understand, the associations could not be shown in the diagram of Figure 8.

The sequence diagram for this scenario, generated by the USE tool, is shown in Figure 9. The scenario exercises all all operations and notification types in the system.

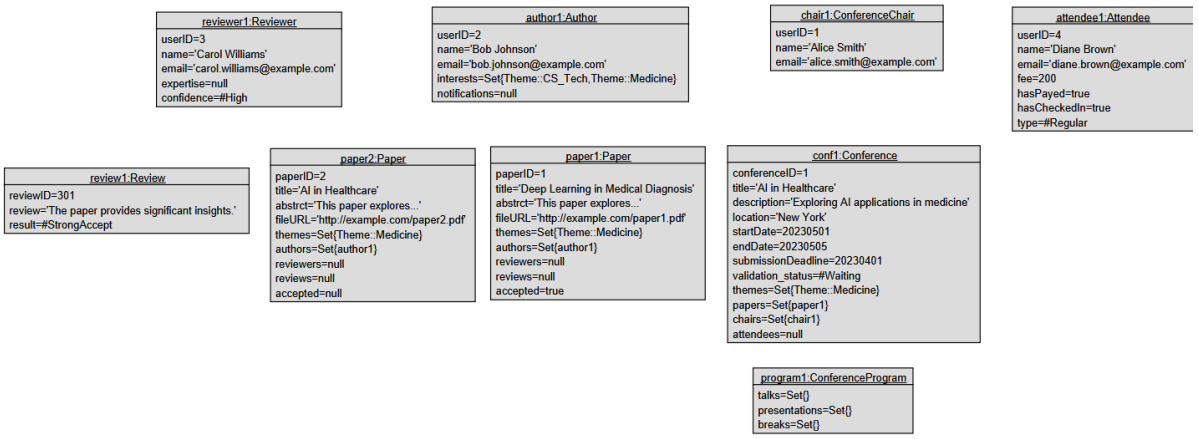


Figure 8. Class diagram generated with the USE tool from the given specification.

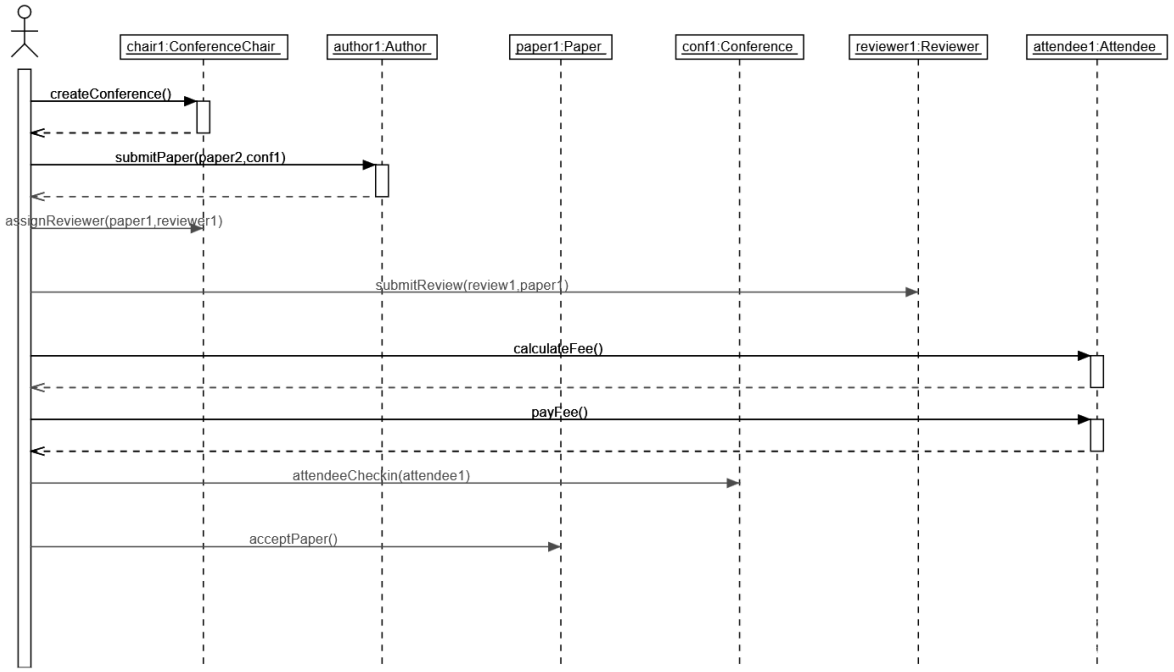


Figure 9. Sequence diagram generated with the USE tool for the given test scenario.

7. Conclusion

The target system (EasyPoltrona) was successfully modelled and specified in UML and OCL, despite some tool limitations.

The contents of this document can be used as a basis for subsequent system development.

This work took approximately two weeks and 30 work hours to develop and document.

Appendix A - Formal specification in OCL/USE

```
model SCMS

-- Class Definitions
class User
      attributes
      userID : Integer
      name : String
      email : String
      constraints
      inv I01_UniqueUserID:
            User.allInstances()->isUnique(u | u.userID)
      inv I02_UniqueValidEmail:
            User.allInstances()->isUnique(email) and
            self.email.indexOf('@') > 0 and
            self.email.indexOf('.') > 0
      inv I03_ValidName:
            self.name.size() > 0
end

class Reviewer < User
      attributes
      expertise : Set(Theme)
      confidence : ReviewerConfidence
      operations
      submitReview(review : Review, paper : Paper)
            pre :
                  paper.reviewers->includes(self) and
                  Paper.allInstances()->includes(paper)
            post :
                  paper.reviews->includes(review)
      constraints
      inv I04_ValidCondfidence:
            self.confidence.oclIsTypeOf(ReviewerConfidence)
      inv I05_ReviewAssignedPapers:
            self.paper->forAll(p | p.reviewers->includes(self))
end

class Author < User
      attributes
      interests : Set(Theme)
      operations
      submitPaper(paper : Paper, conference : Conference)
            pre :
                  Conference.allInstances()->includes(conference) and
                  paper.title.size() > 0 and
```

```
                paper.abstrct.size() > 0 and
                paper.fileURL.size() > 0 and
                paper.authors->includes(self)
          post :
                conference.papers->includes(paper)
    constraints
    inv I06_InterestsValid:
          self.interests->forAll(i | i.oclIsTypeOf(Theme))
    inv I07_SubmitAuthoredPaper:
          Paper.allInstances()->forAll(p |
p.authors->includes(self))
end

class Attendee < User
    attributes
    fee : Real
    hasPayed : Boolean
    hasCheckedIn : Boolean
    type : AttendeeType
    operations
    calculateFee() : Real
          pre :
                self.type.oclIsTypeOf(AttendeeType)
          post :
                let calculatedFee : Real = 200 in
                self.fee = calculatedFee and
                self.fee > 0
    payFee()
          pre :
                self.hasPayed = false
          post :
                self.hasPayed = true
    constraints
    inv I08_ValidAttendeeType:
          self.type.oclIsTypeOf(AttendeeType)
    inv I09_ValidFee:
          self.fee > 0
    inv I10_CheckInWhenPayed:
          self.hasCheckedIn implies self.hasPayed
end

class ConferenceChair < User
    operations
    createConference()
          pre :
                self.oclIsTypeOf(ConferenceChair)
          post :
```

```
                let newConference : Conference =
Conference.allInstances()->any(c | c.conferenceID =
Conference.allInstances()->size()) in
                newConference <> null and
                newConference.validation_status =
ConferenceStatus::Waiting
      assignReviewer(paper : Paper, reviewer : Reviewer)
            pre :
                Reviewer.allInstances()->includes(reviewer) and
                reviewer.confidence <> ReviewerConfidence::None and
                Conference.allInstances().exists(c |
c.papers->includes(paper)) and
                paper.themes->forAll(t |
reviewer.expertise->includes(t))
            post :
                paper.reviewers->includes(reviewer)
end

class Conference
      attributes
      conferenceID : Integer
      title : String
      description : String
      location : String
      startDate : Real
      endDate : Real
      submissionDeadline : Real
      validation_status : ConferenceStatus
      themes : Set(Theme)
      papers : Set(Paper)
      chairs : Set(ConferenceChair)
      attendees : Set(Attendee)
      operations
      setStatus(status : ConferenceStatus)
            pre :
                status.oclIsTypeOf(ConferenceStatus) and
                validation_status = ConferenceStatus::Waiting
            post:
                validation_status = status
      addTheme(theme : Theme)
            pre :
                theme.oclIsTypeOf(Theme)
            post :
                themes->includes(theme)
      addPaper(paper : Paper)
            pre :
                Paper.allInstances()->includes(paper)
            post :
```

```
                papers->includes(paper)
        attendeeCheckin(attendee : Attendee)
                pre :
                        attendees->includes(attendee) and
                        attendee.hasPayed = true
                post :
                        attendee.hasCheckedIn = true
        constraints
        inv I11_UniqueConferenceID:
                Conference.allInstances()->isUnique(c | c.conferenceID)
        inv I12_NonEmpty:
                self.title.size() > 0 and
                self.description.size() > 0 and
                self.location.size() > 0
        inv I13_StartBeforeEnd:
                self.startDate < self.endDate
        inv I14_SubmissionBeforeConference:
                self.submissionDeadline < self.startDate
        inv I15_NonEmptyPaperList:
                self.papers->size() > 0
        inv I16_NonEmptyThemeList:
                self.themes->size() > 0
        inv I17_NonEmptyChairList:
                self.chairs->size() > 0
end

class Paper
        attributes
        paperID : Integer
        title : String
        abstrct : String
        fileURL : String
        themes : Set(Theme)
        authors : Set(Author)
        reviewers : Set(Reviewer)
        reviews : Set(Review)
        accepted : Boolean
        operations
        addReviewer(reviewer : Reviewer)
                pre :
                        self.themes.exists(t |
reviewer.expertise->includes(t)) and
                        not self.reviewers->includes(reviewer)
                post :
                        self.reviewers->includes(reviewer)
        addTheme(theme : Theme)
                pre :
                        theme.oclIsTypeOf(Theme) and
```

```
                not self.themes->includes(theme)
          post :
                self.themes->includes(theme)
     acceptPaper()
          pre :
                self.reviews->select(r |
                                     r.result =
ReviewResult::StrongAccept or
                                     r.result = ReviewResult::Accept or
                                     r.result = ReviewResult::WeakAccept
                                     )->size() >= 3
          post :
                self.accepted = true
     constraints
     inv I18_UniquePaperID:
          Paper.allInstances()->isUnique(p | p.paperID)
     inv I19_NonEmpty:
          self.title.size() > 0 and
          self.abstrct.size() > 0 and
          self.fileURL.size() > 0
     inv I20_NonEmptyThemeList:
          self.themes->size() > 0
     inv I21_NonEmptyAuthorList:
          self.authors->size() > 0
end

class Review
     attributes
     reviewID : Integer
     review : String
     result : ReviewResult
     constraints
     inv I22_UniqueReviewID:
          Review.allInstances()->isUnique(r | r.reviewID)
     inv I23_ValidResult:
          self.result.oclIsTypeOf(ReviewResult)
end

class ConferenceProgram
     attributes
     talks : Set(Tuple(time: Real, speakers: String))
     presentations : Set(Tuple(time: Real, speaker: String))
     breaks : Set(Tuple(time: Real, type: String))
     operations
     generateSchedule()
     constraints
     inv I24_FirstTupleUnique:
```

```
            self.talks->forAll(t | self.presentations->forAll(p | t
<> p)) and
            self.talks->forAll(t | self.breaks->forAll(b | t <> b))
and
            self.presentations->forAll(p | self.breaks->forAll(b | p
<> b))
      inv I25_ValidTimes:
            self.talks->forAll(t | t.time > 0) and
            self.presentations->forAll(p | p.time > 0) and
            self.breaks->forAll(b | b.time > 0)
end

-- Enumeration Definitions
enum ConferenceStatus { Waiting, Approved, Denied }
enum Theme { CS_Tech, Medicine, Sustainability, Physics,
SocialSciences, Engineering }
enum ReviewerConfidence { Expert, High, Medium, Low, None }
enum ReviewResult { StrongAccept, Accept, WeakAccept, Borderline,
WeakReject, Reject, StrongReject }
enum AttendeeType { Regular, Student, Author }

-- Association Definitions
association AuthorPaper between
      Author[1..*]
      Paper[1..*]
end

association AuthorConference between
      Author[*] role notified
      Conference[*] role notifies
end

association ReviewerPaper between
      Reviewer[*]
      Paper[1..*]
end

association ConferencePapers between
      Conference[1]
      Paper[*]
end

association ConferenceChairs between
      Conference[*]
      ConferenceChair[1..*]
end

association ConferenceAttendees between
```

```
      Conference[1]
      Attendee[*]
end

association ConferenceProgramConference between
      Conference[1]
      ConferenceProgram[1]
end

association ReviewerReview between
      Reviewer[1]
      Review[*]
end

association PaperReview between
      Paper[1]
      Review[*]
end
```

Appendix B - Test script in OCL/USE

```
-- Test Case 1: Create a Conference Chair
!create chair1 : ConferenceChair
!set chair1.userID := 1
!set chair1.name := 'Alice Smith'
!set chair1.email := 'alice.smith@example.com'

-- Check constraints for ConferenceChair
check


-- Test Case 2: Create an Author and a Paper
!create author1 : Author
!set author1.userID := 2
!set author1.name := 'Bob Johnson'
!set author1.email := 'bob.johnson@example.com'
!set author1.interests := Set{Theme::CS_Tech, Theme::Medicine}

-- Create a Paper
!create paper1 : Paper
!set paper1.paperID := 1
!set paper1.title := 'Deep Learning in Medical Diagnosis'
!set paper1.abstrct := 'This paper explores...'
!set paper1.fileURL := 'http://example.com/paper1.pdf'
!set paper1.themes := Set{Theme::Medicine}
!set paper1.authors := Set{author1}
```

```
-- Test Case 3: Create a Conference

-- Create a Conference
!openter chair1 createConference()
!create conf1 : Conference
!set conf1.conferenceID := 1
!set conf1.title := 'AI in Healthcare'
!set conf1.description := 'Exploring AI applications in medicine'
!set conf1.location := 'New York'
!set conf1.startDate := 20230501
!set conf1.endDate := 20230505
!set conf1.submissionDeadline := 20230401
!set conf1.validation_status := ConferenceStatus::Waiting
!set conf1.themes := Set{Theme::Medicine}
!set conf1.papers := Set{paper1}
!set conf1.chairs := Set{chair1}
!opexit

-- Check constraints for Author, Paper, Conference
check

-- Add New Paper
!create paper2 : Paper
!set paper2.paperID := 2
!set paper2.title := 'AI in Healthcare'
!set paper2.abstrct := 'This paper explores...'
!set paper2.fileURL := 'http://example.com/paper2.pdf'
!set paper2.themes := Set{Theme::Medicine}
!set paper2.authors := Set{author1}

!openter author1 submitPaper(paper2, conf1)
!insert paper2 into conf1.papers
!opexit

-- Check constraints for Conference
check


-- Test Case 4: Assign a Reviewer to a Paper
!create reviewer1 : Reviewer
!set reviewer1.userID := 3
!set reviewer1.name := 'Carol Williams'
!set reviewer1.email := 'carol.williams@example.com'
!insert Theme::Medicine into reviewer1.expertise
!set reviewer1.confidence := ReviewerConfidence::High

-- Check constraints for Reviewer
```

```
check

-- Assign the Reviewer
!openter chair1 assignReviewer(paper1, reviewer1)
!insert reviewer1 into paper1.reviewers
!opexit

-- Check constraints for Paper-Reviewer association
check

-- Test Case 5: Submit a Review
!create review1 : Review
!set review1.reviewID := 301
!set review1.review := 'The paper provides significant insights.'
!set review1.result := ReviewResult::StrongAccept

!openter reviewer1 submitReview(review1, paper1)
!insert review1 into paper1.reviews
!insert review1 into reviewer1.reviews
!opexit

-- Check constraints for Review
check

-- Test Case 6: Register an Attendee
!create attendee1 : Attendee
!set attendee1.userID := 4
!set attendee1.name := 'Diane Brown'
!set attendee1.email := 'diane.brown@example.com'
!set attendee1.type := AttendeeType::Regular
!set attendee1.fee := 0
!set attendee1.hasPayed := false
!set attendee1.hasCheckedIn := false

-- Calculate Fee
!openter attendee1 calculateFee()
!set attendee1.fee := 200
!opexit

-- Pay Fee
!openter attendee1 payFee()
!set attendee1.hasPayed := true
!opexit

-- Check-in
!openter conf1 attendeeCheckin(attendee1)
!set attendee1.hasCheckedIn := true
!opexit
```

```
-- Check constraints for Attendee
check

-- Test Case 7: Accept a Paper
!openter paper1 acceptPaper()
!set paper1.accepted := true
!opexit

-- Check constraints for Paper Acceptance
check

-- Test Case 8: Conference Program Generation
!create program1 : ConferenceProgram
-- Initialize collections
!set program1.talks := Set{}
!set program1.presentations := Set{}
!set program1.breaks := Set{}

-- Add elements to the collections
!insert Tuple{9.00, 'Keynote: AI in Healthcare'} into program1.talks
!insert Tuple{10.00, 'Deep Learning in Medical Diagnosis'} into
program1.presentations
!insert Tuple{11.00, 'Coffee Break'} into program1.breaks
-- Check constraints for ConferenceProgram
check
```

References

1. UML specification home page: https://www.omg.org/spec/UML/
2. OCL specification home page: https://www.omg.org/spec/OCL/
3. Enterprise Architect home page: https://sparxsystems.com/
4. USE home page: https://sourceforge.net/projects/useocl/
5. Domain model, in Patterns of Enterprise Application Architecture, by Martin Fowler, https://martinfowler.com/eaaCatalog/domainModel.html