UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Departamento de Informática Aplicada

Técnicas de Construção de Programas

INF01120

Prof. Ingrid Nunes

Etapa 2 DESIGNER

GRUPO 01

Turma A

Andréa Leonice dos Santos 00275624

Giusepe Tessari Lopes de Oliveira 00282772

Maria Flávia Borrajo Tondo 00278892

Vanessa Righi Coelho 00243463

1. Domínio

Começamos modelando o domínio da aplicação. Primeiramente, as classes para o pesquisador (`Researcher`), artigo (`Article`) e conferência (`Conference`) foram modeladas. Além do desejo de manter os dados padronizados, seguindo a descrição, durante a definição dos atributos e métodos de cada classe, percebemos que existiam atributos que seriam usados em comparações no decorrer das operações, por isso decidimos criar classes específicas para afiliação (`Affiliation`) e tópicos de pesquisa (`SearchTopics`). A classe `Researcher` contém, além dos atributos explicitados no início da descrição do trabalho, uma lista com os artigos a serem revisados pelo pesquisador, da mesma forma que a classe `Article` contém uma lista de revisores do artigo. Ainda na classe `Article`, os seguinte métodos devem ser salientados:

- `calculateAverage()`, que retorna a média das notas atribuídas ao artigo;
- `hasReceivedAllGrades()`, que retorna se já foram atribuídas todas as notas ao artigo;
- `saveGrade(grade : Grade)`, que salva na lista de notas a avaliação de um revisor para o artigo.

*A classe `Grade` foi criada para que a relação entre nota e revisor pudesse ser facilitada. Dessa forma, cada artigo armazena uma lista com as notas atribuídas a ele, bem como os respectivos pesquisadores que atribuíram cada uma das notas.

- `checkArticleStatus()`, que verifica, a partir da média, se o artigo foi aceito ou rejeitado.

Já na classe `Conference`, devemos mencionar os métodos

- `isConferenceAllocated()`, que verifica se a conferência já foi alocada;
- `allArticlesHaveGrades()`, que verifica se todas as notas já foram atribuídas para os artigos da conferência.

As funcionalidades requisitadas na descrição são responsabilidade das classes `AllocReviewers`, `GradeAssignment` e `ArticlesSelection`. Tais classes estão associadas à `Commands` (`AllocReviewersCommand`, `GradeAssignmentCommand` e `ArticlesSelectionCommand`), que por sua vez são responsáveis por separar a lógica da aplicação da interface com o usuário. Um maior detalhamento será feito mais adiante no relatório.

Na aplicação, a instanciação de `AllocReviewers`, `GradeAssignment` e `ArticlesSelection` será feita em `SystemOperationsImpl`. Essa mesma classe conterá a base de dados, e deverá ser passada como parâmetro na instanciação dos comandos, assegurando assim que eles acessem sempre a mesma database. Tanto `SystemOperationsImpl` quanto a Database são instanciados uma única vez, no construtor de `PeerReviewSystem`.

2. Base de dados

A classe que simula o banco de dados guarda os pesquisadores (`Researcher`), artigos (`Article`) e conferências (`Conference`) através de mapas, utilizando a id como chave.

A função `Database.initData()` é responsável por inicializar as informações do banco de dados, ou seja, adicionar os objetos de forma hard-coded. Isso deve ser feito através dos overloads da função *save*: `Database.save(article : Article)`, `Database.save(researcher : Researcher)` e `Database.save(conference : Conference)`.

A classe possui getters para retornar objetos específicos de acordo com a id

`Database.getConference(ConferenceID : int) : Conference`

`Database.getArticle(ArticleID : int) : Article` e

`Database.getResearcher(ReseracherID : int) : Researcher`

bem como getters usados para retornar todos os objetos de uma determinada classe armazenados na base de dados:

`Database.getAllConferences() : List<Conference>`

`Database.getAllArticles() : List<Article>` e

`Database.getAllResearchers() : List<Researcher>`

Além disso, também possui a função `Database.getNotAllocatedConferences() : List<Conference>`, que é responsável por retornar uma lista com as conferências salvas na base de dados que ainda não foram alocadas. O uso e implementação desta função está melhor detalhado na explicação da funcionalidade de alocação de artigos a membros do comitê de programa, descrita mais adiante no relatório.

3. Interface

Modelado o domínio da aplicação, partimos para o sistema como um todo e a interface com o usuário.

A classe `PeerReviewSystem` contém o método `main()`, e é onde o programa é inicializado. Essa classe tem como finalidade mostrar um menu (textual) para o usuário, informando as três funcionalidades disponíveis no programa, além de criar, em sua instanciação, a base de dados e o objeto *systemOperations*. Além disso, ela contém um atributo chamado *commands*, do tipo `Map<String,UIAction>`, que no construtor deve ser preenchido com os comandos disponíveis através da função `addCommand(key : String, command : Command)`. O método `createAndShowUI()` é responsável por mostrar o menu de comandos e obter o comando escolhido pelo usuário. Já a função `getMenu()` deverá construir o menu a partir do mapa de *commands*, e será chamada dentro de `createAndShowUI()`.

Para fazer a criação do objeto *systemOperations*, do tipo `SystemOperationsImpl`, é necessário que a base de dados instanciada seja passada como argumento. Esse objeto será usado para instanciar os comandos após o usuário selecionar qual a operação desejada. *systemOperations* é definido como *protected* na classe abstrata `Command` para que todas as classes que a estendem sejam capazes de acessá-lo.

A comunicação efetiva com o usuário é feita através de uma classe auxiliar chamada `UserTextInteraction`, que contém métodos de uso comum para solicitação de dados ao usuário.

Na classe `PeerReviewSystem`, o método `UserTextInteraction.readSelectedCommand()` seria chamado para ler do usuário qual o comando selecionado. Assim que o usuário selecionar uma opção válida, `PeerReviewSystem` consulta seu mapa de comandos e instancia o comando correto (`AllocReviewersCommand`, `ArticlesSelectionCommand` ou `GradeAssignmentCommand`), passando `systemOperations` como parâmetro, e chama o método `execute()` definido em `UIAction`.

A partir da instanciação do `Command` desejado, partimos para as funcionalidades da aplicação.

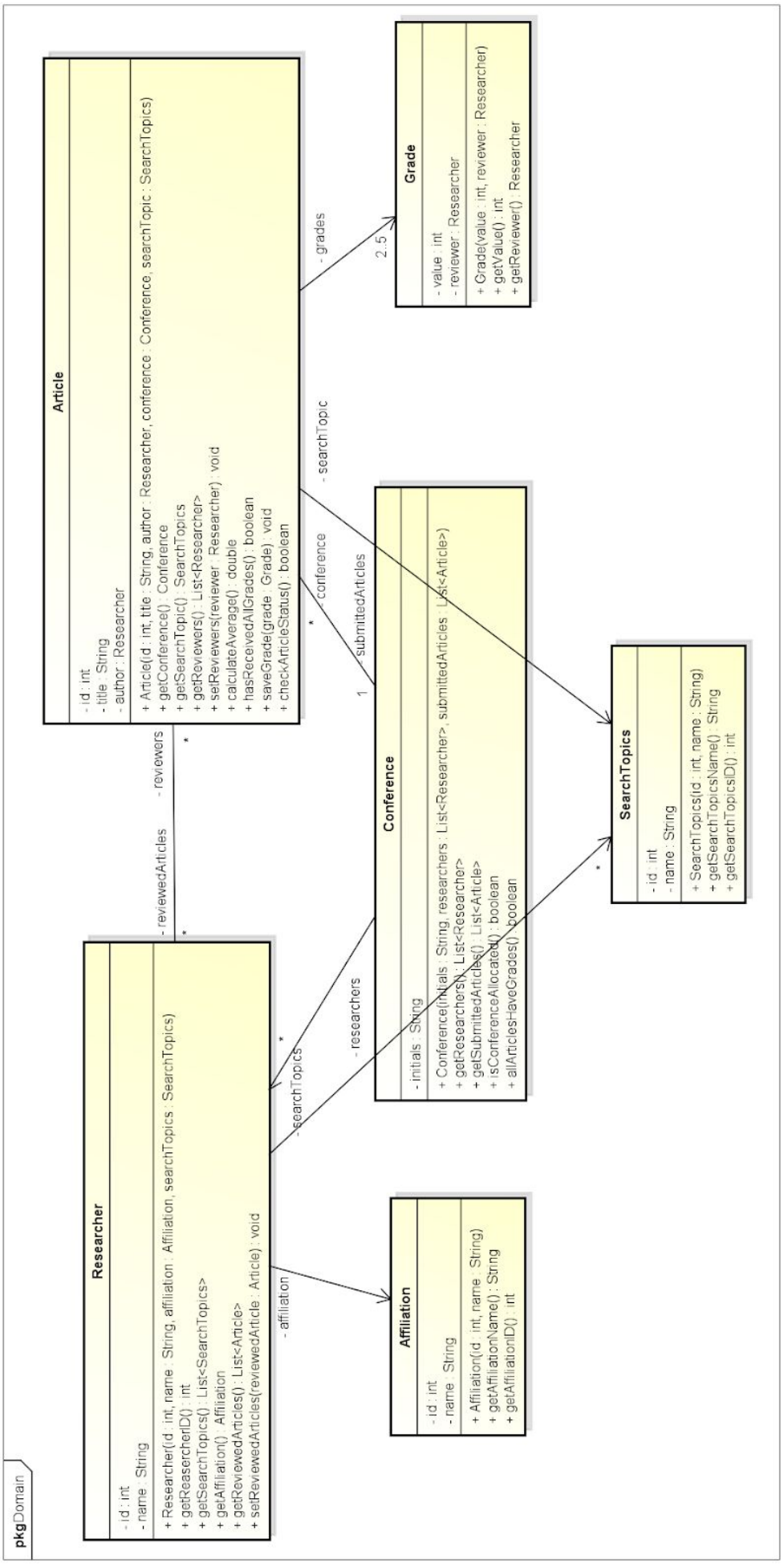

Figura 1 - Diagrama de Classes do Domínio

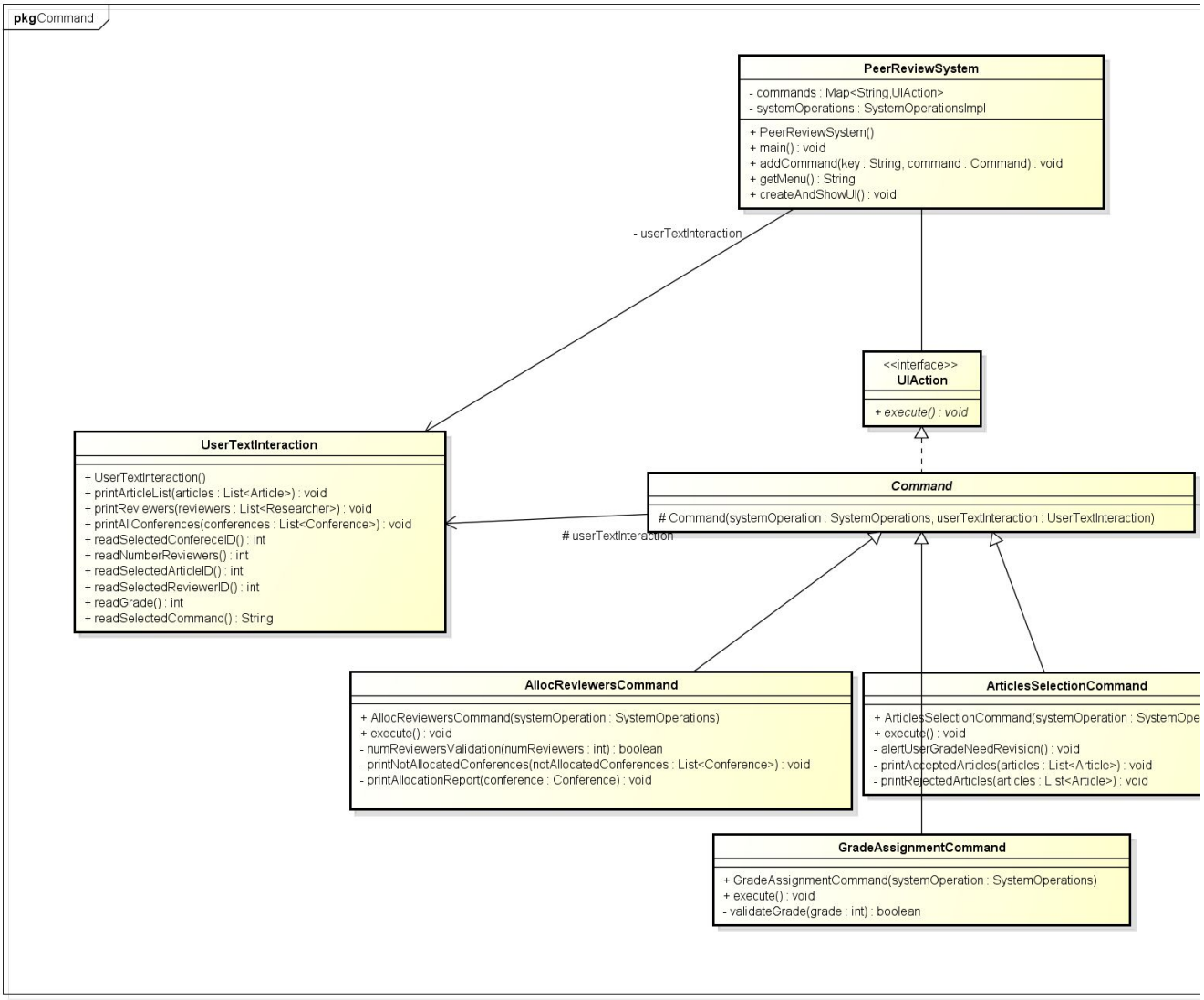

Figura 2 – Diagrama de Classes da Aplicação (Parte 1)

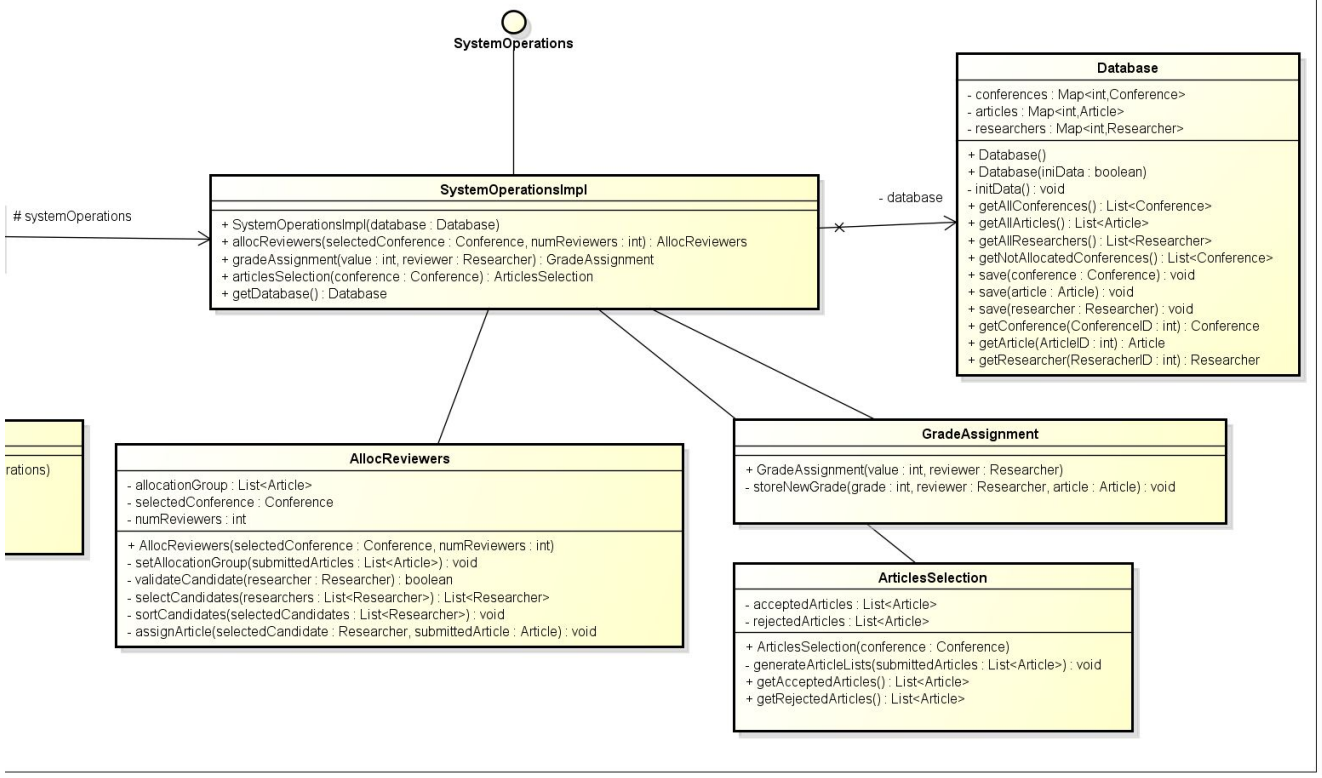

Figura 3 – Diagrama de Classes da Aplicação(Parte 2)

4. Alocação de artigos a Membros do Comitê de Programa

Quando a operação solicitada é a alocação de artigos a membros do comitê do programa, `PeerReviewSystem` instancia `AllocReviewersCommand` e então chama sua função `execute()`.

`AllocReviewersCommand` solicita à base de dados quais conferências ainda não foram alocadas, através de `systemOperations.getDatabase.getNotAllocatedConferences()`, que basicamente verifica, para cada conferência salva na base de dados, se já foi alocada ou não, usando para isso o método `Conference.isConferenceAllocated()`, e retorna uma lista com as conferências ainda não alocadas. Esta lista é exibida para o usuário através de `UserTextInteraction.printNotAllocatedConferences()`. O sistema pede para o usuário informar para qual conferência será feita a alocação, através de `UserTextInteraction.readSelectedConfereceID()`, que retorna a ID da conferência selecionada. Em seguida, o respectivo objeto associado àquela ID é obtido através da database (`systemOperations.getDatabase.getConference(ConferenceID)`). Por fim, o número de revisores, que deve ser entre 2 e 5, é lido através de `UserTextInteraction.readNumberReviewers()`, e a verificação da validade do valor informado é feita a partir da função privada `AllocReviewersCommand.numReviewersValidation(numReviewers)`.

O sistema, então, instancia um objeto do tipo `AllocReviewers` usando `SystemOperationsImpl.allocReviewers()`. `AllocReviewers` é responsável por realizar efetivamente a alocação dos membros; enquanto o número de revisores alocados não for igual ao número de revisores solicitado pelo usuário, `AllocReviewers` reinicializa um conjunto de alocação através da função `AllocReviewers.setAllocationGroup()`, que deve fazer os devidos acessos à conferência para obter os artigos, e, em seguida, seleciona os candidatos através de `AllocReviewers.selectCandidates()`, validando-os de acordo com a especificação pedida no enunciado através de `AllocReviewers.validateCandidate()`. O sistema então ordena esses candidatos pelos critérios pedidos na especificação e, usando `AllocReviewers.sortCandidates()`, atribui a revisão do artigo ao primeiro membro da lista, através de `AllocReviewers.assignArticle()`.

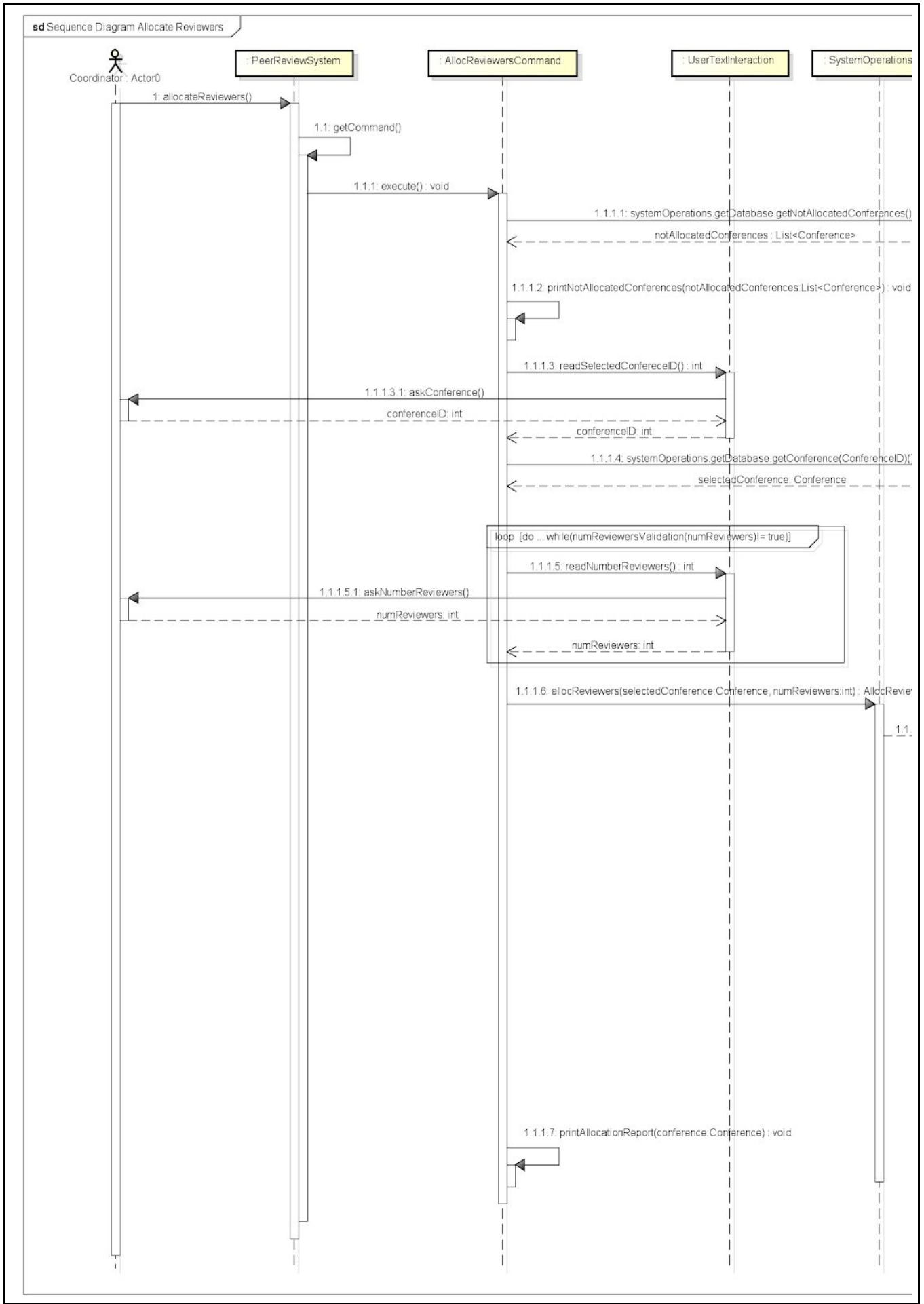

Figura 4 - Diagrama de Sequência Alocação de Artigos a Membros do Comitê do Programa (Parte 1)

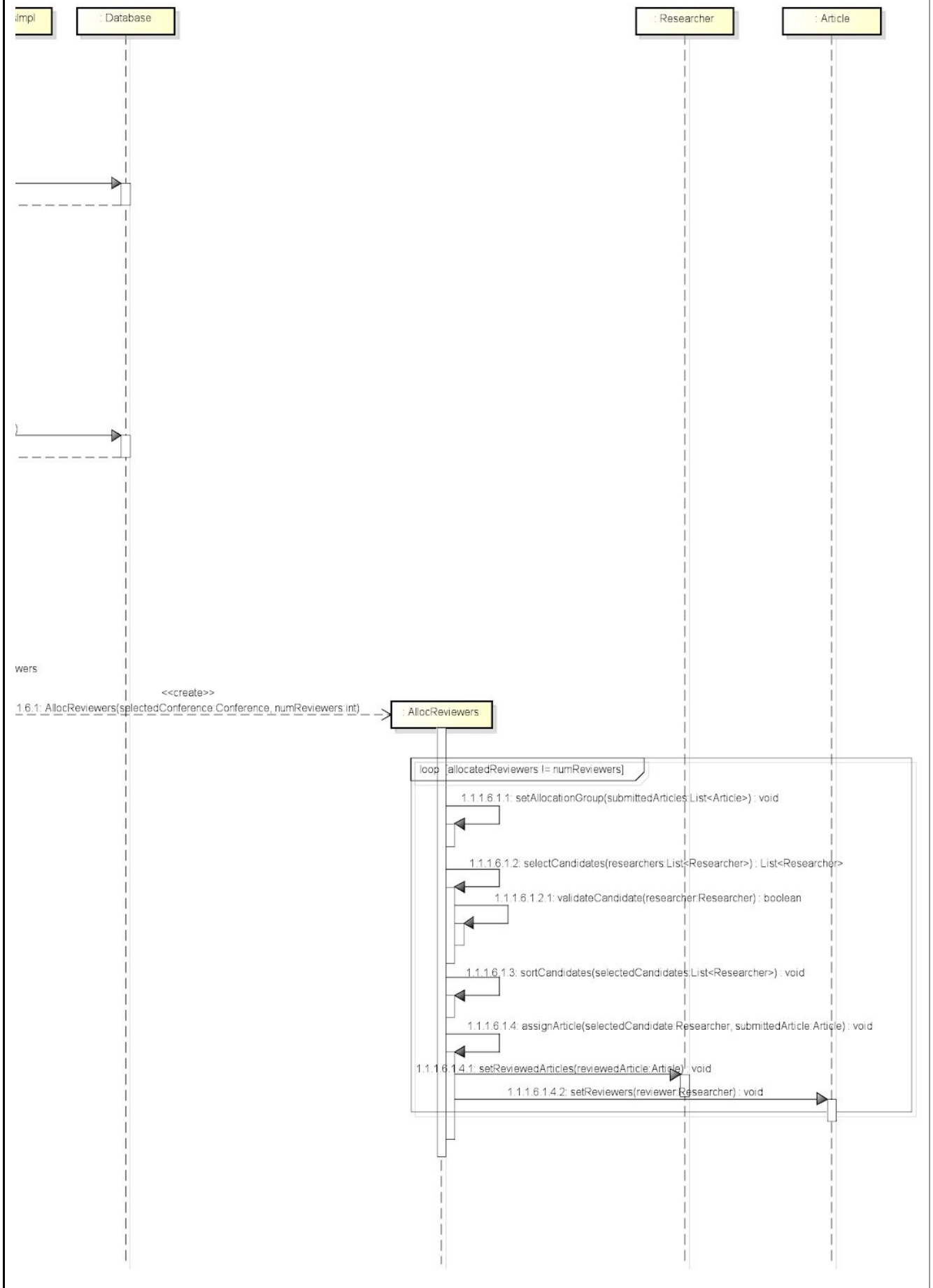

Figura 5 - Diagrama de Sequência Alocação de Artigos a Membros do Comitê do Programa (Parte 2)

5. Atribuição de Notas a Artigos

Quando a operação solicitada é a atribuição de notas a artigos, `PeerReviewSystem` instancia `GradeAssignmentCommand` e então chama sua função `execute()`.

`GradeAssignmentCommand` então solicita à base de dados a lista com todos os artigos, através de `systemOperations.getDatabase.getAllArticles()`, e essa lista é mostrada ao usuário fazendo o uso de `UserTextInteraction.showArticleList()`. `GradeAssignmentCommand` então pede para o usuário informar para qual artigo ele deseja atribuir uma nota, através de `UserTextInteraction.readSelectedArticleID()`, que retorna o ID do artigo selecionado.

O objeto `Article` é solicitado da base de dados a partir do seu ID (`systemOperations.getDatabase.getArticle(articleID)`). A lista de revisores deste artigo, ou seja, a lista apontada por `selectedArticle.getReviewers()`, é mostrada ao usuário usando a função `UserTextInteraction.showReviewers(selectedArticle.getReviewers())`. O sistema, então, faz a leitura do id do revisor escolhido pelo usuário `UserTextInteraction.readSelectedReviewerID()` e solicita à base de dados o revisor em questão através de `systemOperations.getDatabase.getResearcher(reviewerID)`. O sistema pede, através de `UserTextInteraction.readGrade()`, a nota para o artigo. Enquanto o usuário não informa uma nota válida, ou seja, entre -3 e 3, o sistema continua perguntando qual nota deve ser associada ao artigo. Essa verificação é feita através da função privada `GradeAssignmentCommand.validateGrade(grade : int)`, que é do tipo booleana. Informada uma nota válida, o sistema instancia um objeto do tipo `GradeAssignment` para atribuir a nota ao artigo, isto é feito usando `systemOperationsImpl.gradeAssignment(value : int, reviewer : Researcher)`. `GradeAssignment` é responsável por armazenar a nota informada fazendo uso do método `GradeAssignment.storeNewGrade()`, que deve instanciar um objeto do tipo `Grade` com a nota `grade`: `int` e o `reviewer` : `Researcher` recebidos e, então, salvá-lo na lista de notas do artigo, através de `article.saveGrade()`.

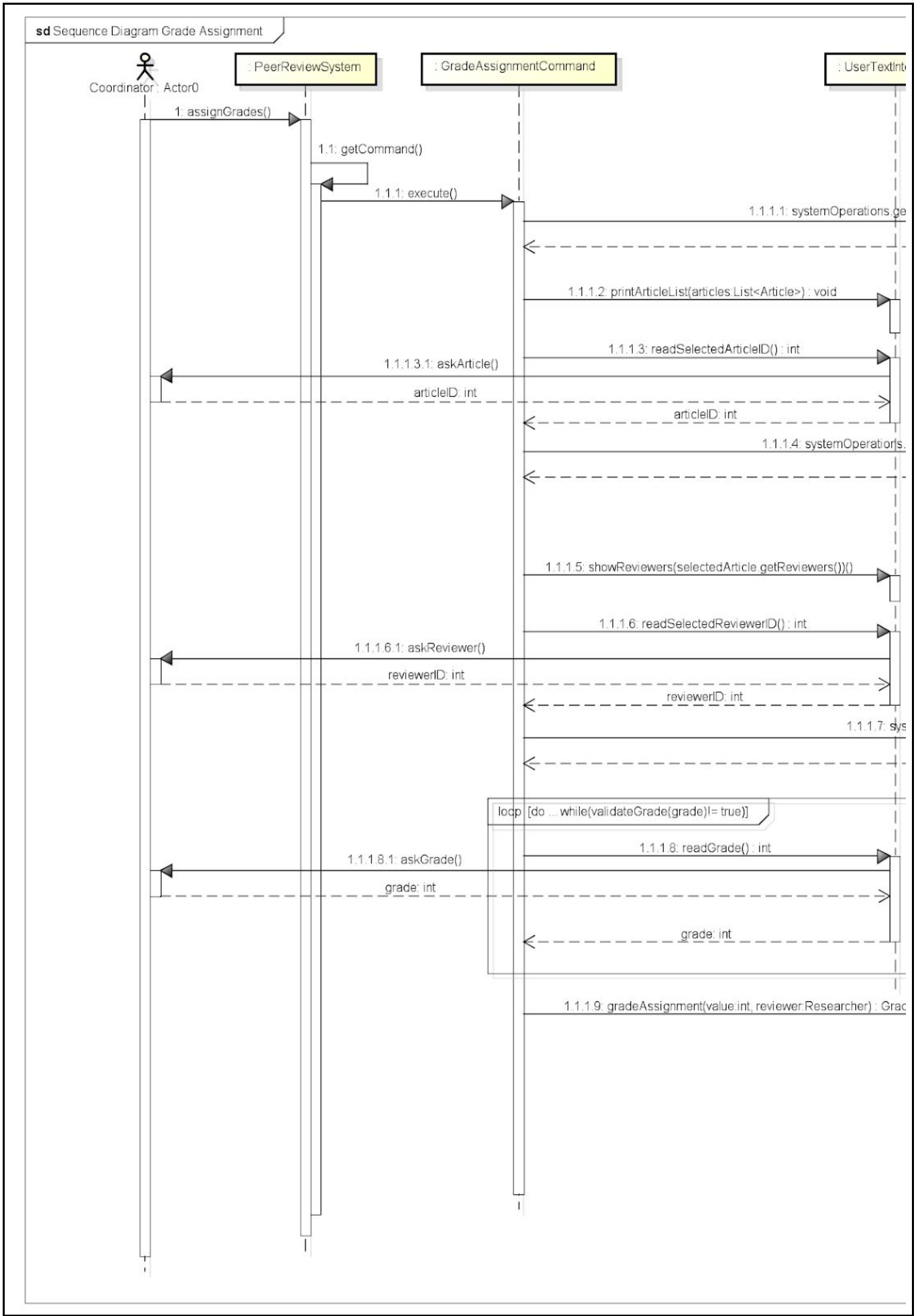

Figura 6 - Diagrama de SequênciaAtribuição de Notas a Artigos (Parte 1)

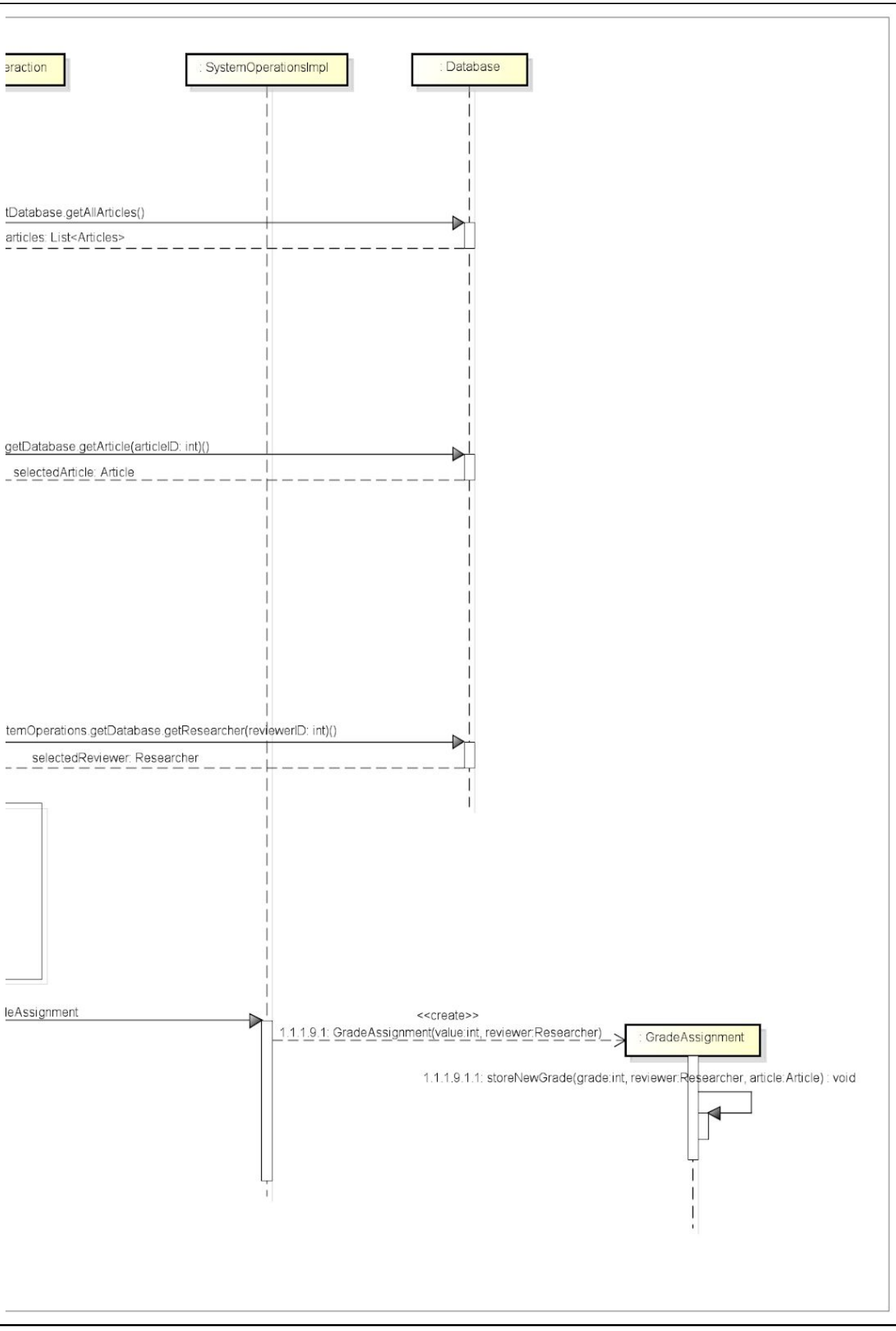

Figura 7 - Diagrama de Sequência Atribuição de Notas a Artigos (Parte 2)

6. Seleção de Artigos

Quando a operação solicitada é a atribuição de notas a artigos, `PeerReviewSystem` instancia `ArticlesSelectionCommand` e então chama sua função `execute()`.

Dentro desta função, a classe `ArticlesSelectionCommand` solicita a base de dados a lista com todas as conferências através de `systemOperations.getDatabase.getAllConferences()`. A lista com todas as conferências é então exibida ao usuário (`UserTextInteraction.showAllConferences())`. É solicitado então de qual conferência o usuário deseja selecionar os artigos, sendo essa comunicação feita através da classe `UserTextInteraction`. Depois de retornada a id de qual conferência o usuário deseja, a conferência é solicitada para a base de dados através de sua ID e a base de dados retorna o objeto da conferência selecionada.

É, então, testado se todos os artigos já receberam notas, através da função `Conference.allArticlesHaveGrades()` que em sua implementação deve checar se cada artigo em sua coleção de `submittedArticles` já recebeu notas de todos os seus revisores. Isso deve ser feito chamando `Article.hasReceivedAllGrades()` que faz exatamente isso: dada sua coleção de `Grades` e de `Reviewers`, checa se já recebeu uma nota de cada revisor.

Caso `Conference.allArticlesHaveGrades()` retorne verdadeiro, um objeto do tipo `ArticlesSelection` é instanciado através de `systemOperationsImpl.articlesSelection(conference : Conference)`. Essa classe é então responsável por verificar quais artigos são aceitos e quais são rejeitados, chamando `ArticlesSelection.generateArticleLists()`.Para isso, esse método chama `Article.checkArticleStatus(),` que por sua vez faz uso do método `Article.calculateAverage()` para cada artigo, preenchendo, assim, suas listas de artigos `acceptedArticles` e `rejectedArticles`.

O controle é retornado para `ArticlesSelectionCommand` que pega as listas de artigos aceitos e rejeitados através de seus getters `ArticlesSelection.getAcceptedArticles()` e `ArticlesSelection.getRejectedArticles()` e as exibe para o usuário através de `userTextInteraction.showArticleList()`.

Caso `Conference.allArticlesHaveGrades()` retorne falso, o usuário é alertado que pelo menos um revisor ainda não atribuiu nota à algum artigo através de `ArticlesSelectionCommand.alertUserGradeNeedRevision()`.

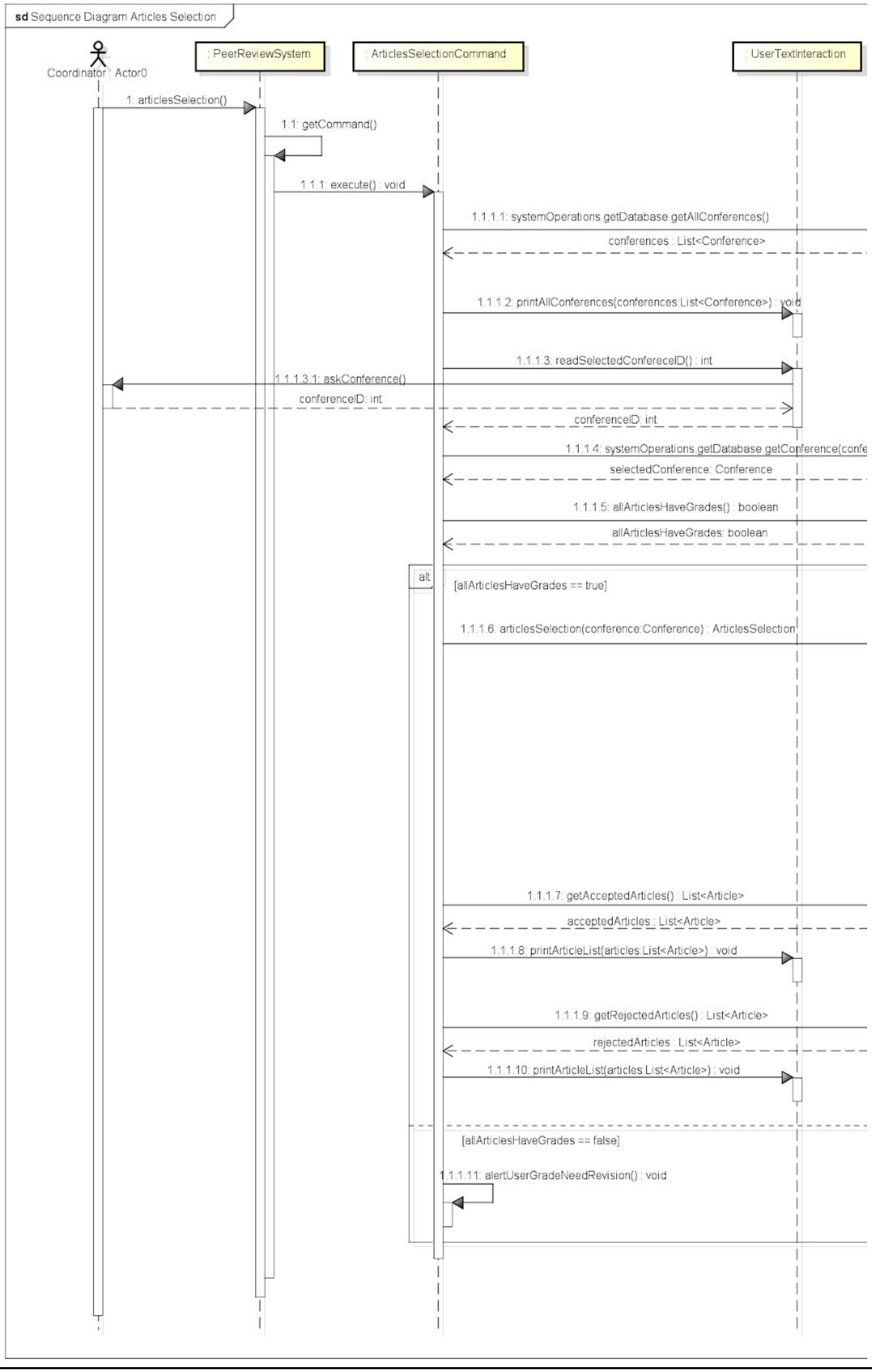

Figura 8 - Diagrama de Sequência Seleção de Artigos (Parte 1)

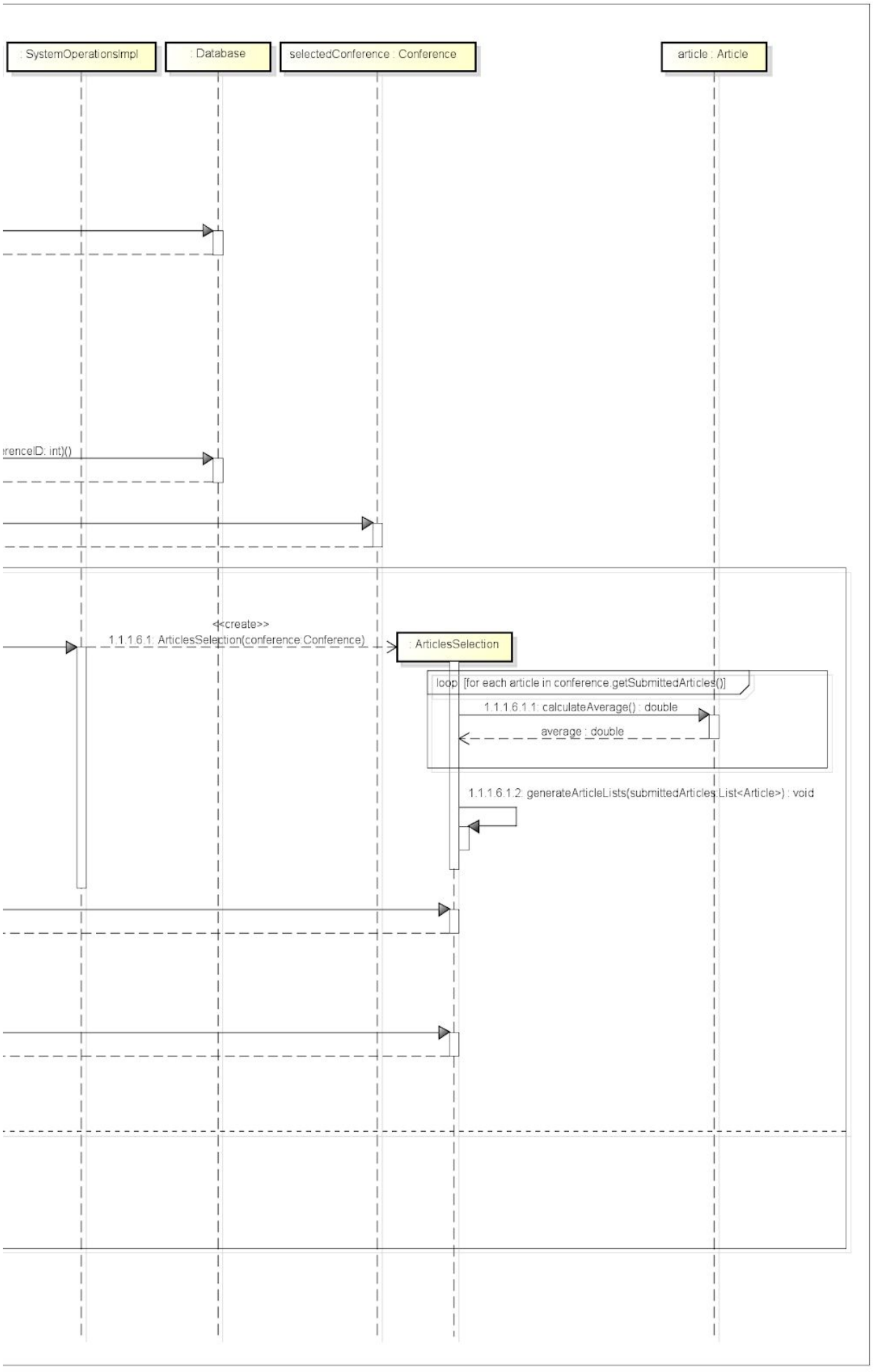

Figura 9 - Diagrama de Sequência Seleção de Artigos (Parte 2)