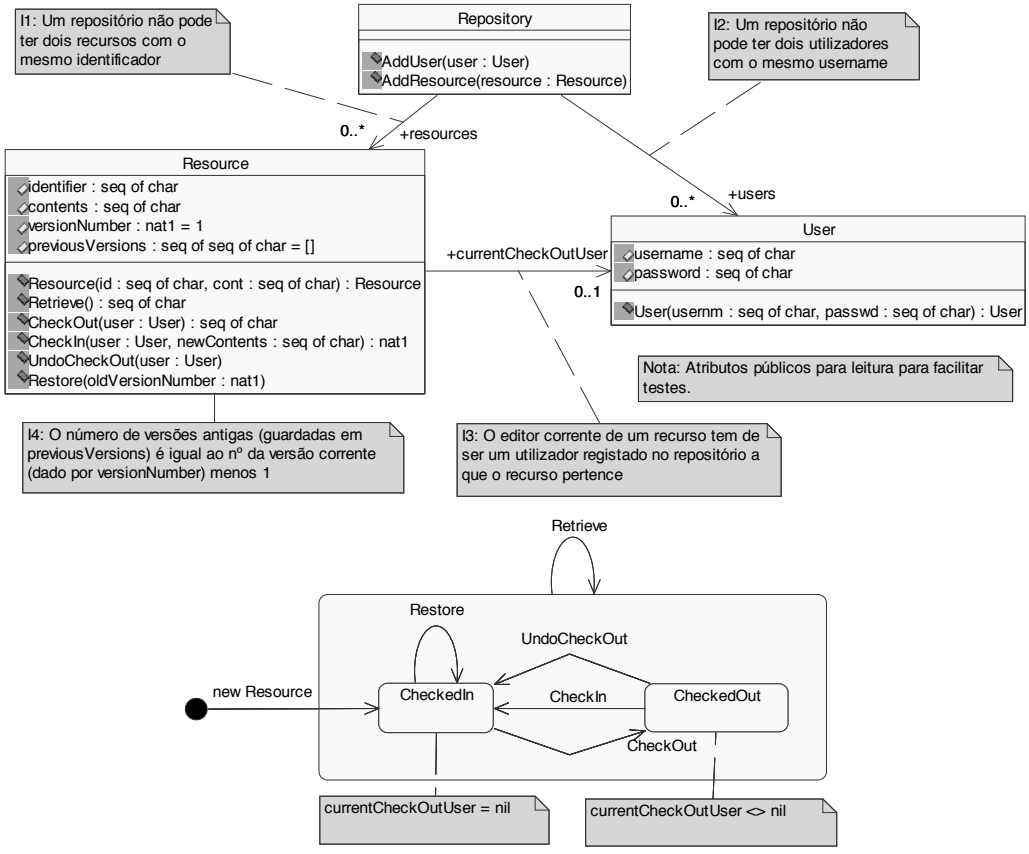
Métodos Formais em Engenharia de Software

1.º Ano de Mestrado de Informática da Universidade do Minho
Ano Lectivo de 2007/08

Prova de avaliação — Prática Laboratorial — 21 de Fevereiro 2008
14h30
Sala 1.08

PROVA COM CONSULTA (3 horas)

Considere o modelo UML de um sistema de controlo de versões básico (com mecanismo de check-in/check-out) apresentado nos dois diagramas seguintes (diagrama de classes do sistema e diagrama de estados de um recurso).



Questão 1 (VDM++)

Os ficheiros `Repository.rtf`, `Resource.rtf` e `User.rtf` contêm o resultado do mapeamento do diagrama de classes UML para VDM++, a que foram acrescentados alguns comentários e a especificação de algumas operações. Existe um método de teste (`TestCheckinCheckout`) já criado em VDM++ no ficheiro `TestRepository.rtf`. Nas VDM Tools, abra o projecto `controlo_versoes.prj` e certifique-se que as verificações de sintaxe e de tipos são bem sucedidas. Em caso de problemas com caminhos de acesso aos ficheiros, remover e voltar a adicionar os ficheiros `Repository.rtf`, `Resource.rtf`, `User.rtf`, `Test.rtf` e `TestRepository.rtf` ao projecto. Certifique-se que nas opções do projecto a verificação de invariantes, pré-condições e pós-condições está activada.

1. Na classe `Resource`, escreva o corpo e pré-condição das operações `CheckIn` e `CheckOut`, de forma consistente com o diagrama de estados e com o método de teste `TestCheckinCheckout`, que no final deve executar com sucesso (isto é, sem quebrar asserções, invariantes ou pré-condições).
2. Complete a especificação da classe `Repository`, com invariantes e pré-condições. Certifique-se que o método de teste `TestCheckinCheckout` continua a executar com sucesso.
3. Na classe `Resource`, escreva o corpo e pré-condição das operações `UndoCheckOut` e `Restore`, de forma consistente com o diagrama de estados e com os comentários fornecidos.

No final, entregar os ficheiros `Resource.rtf` e `Repository.rtf` modificados.

Questão 2 (Alloy)

O ficheiro `Repository.als` contém uma especificação incompleta dste sistema de controlo de versões em Alloy. Note que o identificador deixou de ser um atributo do recurso, para passar a ser o domínio de uma relação que permite aceder aos mesmos.

1. Defina o predicado `Correct_Versions` que testa se as versões de um recurso são válidas: as antigas tem que estar numeradas sequencialmente (sem falhas) e a versão actual tem que ser a próxima na sequência.
2. Defina o predicado `Inv` que testa se um repositório é válido: para além de testar que todos os recursos têm versões válidas, tem que garantir que os utilizadores tem todos identificadores diferentes, e que todos os utilizadores que realizaram *checkouts* estão registados no sistema.
3. Complete a definição dos predicados `add`, `retrieve`, `checkout`, `checkin`, e `restore` por forma a validar as asserções respectivas. Não se esqueça de definir pré-condições por forma a tornar os predicados consistentes com o diagrama de estados apresentado.

Não se esqueça de utilizar comandos `run` para detectar casos de sobre-especificação (uma especificação demasiado restritiva que faz com que o predicado seja sempre falso). No final, entregar o ficheiro `Repository.als` modificado.

Questão 3 (JML)

Os ficheiros `Repository.java`, `Resource.java`, `TestRepository.java` e `User.java` contêm uma implementação em Java deste sistema de controlo de versões.

1. Defina no ficheiro `Resource.java` anotações JML para o invariante da classe e para as pré e pós condições dos métodos, de forma consistente com os diagramas acima apresentados.
2. Defina no ficheiro `Repository.java` anotações JML para os invariantes de classe em falta.

No final, entregar os ficheiros `Resource.java` e `Repository.java` modificados.

Questão 4 (Testing)

1. Create unit tests for the Java code given in the previous question using the JUnit framework. The tests should not just invoke the code, but also contain interesting assert statements. Include tests to cover all transitions of state diagram presented above.
2. Measure the unit test coverage for the given Java code using the EMMA coverage analyzer. Bring the coverage level to an acceptable percentage.

Deliver the produced code in the `Testing` directory.

Questão 5 (Metrics)

Create quality profiles for the FindBugs system. Metric reports have been pre-generated using JavaNCSS for several versions of FindBugs (see spreadsheet `findbugs metrics.xls`).

1. Make quality profiles, using the following guidelines:
   - Complexity (CCN) risk categories: 1-3, 4-10, 11-20, 21-50, 50+.
   - Sum of volumes (NCSS) per risk category.
   - Percentage volume per risk category.

   Create pie-charts or stacked bar of percentages charts.
2. Create profiles for each version. Put the profiles side by side to see trend. Which version is best? Which is worst? Why?

Deliver your answers in a modified spreadsheet `findbugs metrics.xls`.
