

**Scientific Paper Indexer**

*Relatório Final*

Métodos Formais em Engenharia de Software

4º ano do Mestrado Integrado em Engenharia Informática e Computação

Índice

[Lista de requisitos e identificação das principais restrições ao funcionamento correto do sistema de software. 3](#_Toc309811996)

[Especificação, em VDM++, das restrições identificadas. 4](#_Toc309811997)

[Classes e scripts de teste. 5](#_Toc309811998)

[Matriz de rastreabilidade dos testes com os requisitos. 6](#_Toc309811999)

[Diagrama concetual de classes do sistema, em UML, incluindo as principais operações de transações e consultas. 7](#_Toc309812000)

[Definição completa das classes em VDM++, incluindo invariantes, pré-condições e póscondições. 8](#_Toc309812001)

[Informação de cobertura dos testes (ficheiros .rtf.rtf obtidos com pretty print depois de correr os scripts de teste). 9](#_Toc309812002)

[Análise da consistência do modelo. 10](#_Toc309812003)

# Lista de requisitos e identificação das principais restrições ao funcionamento correto do sistema de software.

R1 - Adicionar autor (nome);

R2 - Listar autores;

R3 - Adicionar afiliações (nome);

R4 - Listar afiliações;

R5 - Relacionar autor e afiliação;

R6 - Adicionar publicação (nome, descrição, autores, referências e a afiliação;

R7 - Listar publicações;

R8 - Adicionar interesse a um autor;

R9 - Calcular caminho entre dois autores;

R10 - Calcular citações pelo próprio autor;

R11 - Calcular citações de um autor citado por outros autores;

R12 - Contar publicações de um autor;

R13 - Calcular distância entre dois autores;

R14 - Encontrar autores por afiliação;

R15 - Listar publicações por autor;

R16 – Uma publicação não pode fazer referência a uma outra, se a sua data for anterior à que pretende referir;

R17 – Se uma publicação for referenciada, deve ser adicionada uma referência às citações da publicação referida com a publicação que a referiu, o capítulo e a linha;

R18 – A data tem de ser válida.

# Especificação, em VDM++, das restrições identificadas.

**class** **Reference**

instance variables

public chapter : nat;

public line : nat;

public publication : Publication;

operations

public Reference : nat \* nat \* Publication ==> Reference

Reference(c, l, p) == ( chapter := c; line := l;

publication := p)

post chapter = c and line = l and publication = p;

end Reference

**class** **Affiliation**

instance variables

public name : seq of char;

operations

public Affiliation : seq of char ==> Affiliation

Affiliation(n) == (name := n)

post name = n;

public setName: seq of char ==> ()

setName(n) == (name:=n;)

post name = n;

end Affiliation

**class** **Date**

instance variables

public day : nat1;

public month: nat1;

public year : int;

operations

public Date : nat1 \* nat1 \* int ==> Date

Date(d,m,y) == (day := d; month := m; year := y;)

pre d <=31 and

m <= 12 and

if m in set {4,9,6,11}

then d <= 30

else (m = 2) => (d <= 29)

post day = d and month = m and year = y;

end Date

**class** **Interest**

instance variables

public **name** : seq of char;

operations

public Interest : seq of char ==> Interest

Interest(n) == (name := n)

post name = n;

end Interest

**class** **Publication**

instance variables

public name : seq of char;

public description : seq of char;

public date : Date;

public authors : set of Author;

public affiliation : Affiliation;

public references : set of Reference;

public citations: set of Reference;

operations

public Publication : seq of char \* seq of char \* Date

\* set of Author \* Affiliation

\* set of Reference ==> Publication

Publication(n, des, dat, author, aff, ref) ==

(name := n; description := des; date := dat;

authors := author; affiliation := aff;

references := ref; citations := {};

for all ref in set references do

ref.publication.addCitation(new Reference(ref.chapter, ref.line, self));

)

post name = n and description = des and date = dat

and authors = author and affiliation = aff

and references = ref;

public compareDate: Date \* Date ==> bool

compareDate(d1, d2) == (

if d1.year < d2.year

then return true

else if d1.year = d2.year

and d1.month < d2.month

then return true

else if d1.year = d2.year

and d1.month = d2.month

and d1.day <= d2.day

then return true

else return false;);

public setName: seq of char ==> ()

setName(n) == (name:=n;)

post name = n;

public addCitation : Reference ==> ()

addCitation (r) == (citations := citations union {r}; )

pre compareDate(date, r.publication.date)

post r in set citations;

end Publication

**class** **Author**

instance variables

public name : seq of char;

public interests: set of Interest := {};

operations

public Author : seq of char ==> Author

Author(n) == (name := n;)

post name = n;

public addInterest: Interest ==> ()

addInterest(i) == (interests:= interests union {i};

)

pre i not in set interests

post i in set interests;

public setName: seq of char ==> ()

setName(n) == (name := n)

post name = n;

end Author

**class** **Indexer**

instance variables

public publications: set of Publication;

public authors: map Author to set of Affiliation;

public affiliations : set of Affiliation;

operations

public Indexer : () ==> Indexer

Indexer () ==

(publications:= {}; authors := {|->};);

public addPublication: Publication ==> ()

addPublication(p) == (publications := publications union {p};

)

pre p not in set publications and

p.authors subset dom authors and forall ref in set p.references & ref.publication

in set publications

post p in set publications;

public countPublicationsByAuthor : Author ==> real

countPublicationsByAuthor(a) ==

(dcl count : real := 0;

for all pub in set publications do

if a in set pub.authors then count := count + 1;

return count;)

pre a in set dom authors;

public countCitationsByOthers : Author ==> real

countCitationsByOthers (a) ==

(dcl count : real := 0;

for all pub in set publications do

if a not in set pub.authors then

for all ref in set pub.references do

if a in set ref.publication.authors

then count := count + 1;

return count;)

pre a in set dom authors;

public countCitationsByMyself : Author ==> real

countCitationsByMyself (a) ==

(dcl count : real := 0;

for all pub in set publications do

if a in set pub.authors then

for all ref in set pub.references do

if a in set ref.publication.authors

then count := count + 1;

return count;)

pre a in set dom authors;

public coAuthorPath : Author ==> set of Author

coAuthorPath(a) ==

(dcl auths : set of Author := {};

for all pub in set publications do

if a in set pub.authors then

auths := auths union pub.authors;

auths := auths \ {a}; return auths;)

pre a in set dom authors;

public distanceBetween : Author \* Author ==> real

distanceBetween(a1, a2) == return distanceBetweenAux (a1, a2, 0, {a1});

public distanceBetweenAux : Author \* Author \* real \* set of Author ==> real

distanceBetweenAux (a1, a2, sum, visited) ==

( for all pub in set publications do

if a1 in set pub.authors and a2 in set pub.authors then

(return sum + 1;);

for all pub in set publications do

if a1 in set pub.authors then

for all a3 in set pub.authors do if a3 <> a2 and a3 <> a1 and a3 not in set visited

then (distanceBetweenAux(a3, a2, sum + 1, visited union {a3}););

return 0;

)

pre a1 <> a2 and a1 in set dom authors and a2 in set dom authors;

public pathBetween : Author \* Author ==> seq of Author

pathBetween(a1, a2) == return pathBetweenAux(a1,a2,[a1]);

public pathBetweenAux: Author \* Author \* seq of Author ==> seq of Author

pathBetweenAux (a1, a2, visited) ==

(

for all pub in set publications do

if a1 in set pub.authors and a2 in set pub.authors then

(return visited ^ [a2];);

for all pub in set publications do

if a1 in set pub.authors then

for all a3 in set pub.authors do if a3 <> a2 and a3 <> a1 and a3 not in set elems visited

then (pathBetweenAux(a3, a2, visited ^ [a3]););

return [];

)

pre a1 <> a2 and a1 in set dom authors and a2 in set dom authors;

public addAuthor: Author ==> ()

addAuthor(a) == (authors := authors munion {a|->{}})

pre a not in set dom authors

post a in set dom authors;

public addAffiliation: Author \* Affiliation ==> ()

addAffiliation(a, aff) == (authors(a) := authors(a) union {aff};)

pre a in set dom authors

post aff in set authors(a);

public getAuthorsByAff: Affiliation ==> set of Author

getAuthorsByAff (aff) == (

dcl ret : set of Author := {};

for all a in set dom authors do if aff in set authors(a) then ret := ret union {a};

return ret;);

public getPublicationsByAuthor: Author ==> set of Publication

getPublicationsByAuthor(a) == (

dcl ret : set of Publication := {};

for all p in set publications do

if a in set p.authors then ret := ret union {p}; return ret;)

pre a in set dom authors;

end Indexer

# Classes e scripts de teste.

# Matriz de rastreabilidade dos testes com os requisitos.

# Diagrama concetual de classes do sistema, em UML, incluindo as principais operações de transações e consultas.

# Definição completa das classes em VDM++, incluindo invariantes, pré-condições e póscondições.

# Informação de cobertura dos testes (ficheiros .rtf.rtf obtidos com pretty print depois de correr os scripts de teste).

# Análise da consistência do modelo.