

**Scientific Paper Indexer**

*Relatório Final*

Métodos Formais em Engenharia de Software

4º ano do Mestrado Integrado em Engenharia Informática e Computação

Carlos Tiago da Rocha Babo – ei08118

Hélder Alexandre dos Santos Moreira – ei08170

Índice

[Introdução 3](#_Toc310173576)

[Lista de requisitos e principais restrições 4](#_Toc310173577)

[Especificação das classes 5](#_Toc310173578)

[Análise das restrições: 9](#_Toc310173579)

[Matriz de rastreabilidade dos testes com os requisitos e restrições 11](#_Toc310173580)

[Diagrama conceptual de classes do sistema 12](#_Toc310173581)

[Informação de cobertura dos testes 13](#_Toc310173582)

[Análise da consistência do modelo 15](#_Toc310173583)

[Estimativa do tempo gasto 18](#_Toc310173584)

[Conclusão 19](#_Toc310173585)

# Introdução

Este trabalho foi desenvolvido no âmbito da cadeira Métodos Formais em Engenharia de Software e teve como objetivo a aplicação prática dos conceitos de modelação formal de um problema aprendidos na cadeira. Para isso foi desenvolvido na linguagem VDM, na sua versão orientada a objetos, VDM++ e utilizada a ferramenta sugerida, VDM++ Toolbox.

O projeto implementa um indexador de publicações, uma versão simplificada e semelhante ao que pode ser encontrado aqui: <http://academic.research.microsoft.com/> . Na secção seguinte é possível encontrar uma lista de requisitos e restrições que o programa respeita e que são o centro da implementação. Foi também implementada um interface de texto em Java (usando o código gerado para as classes gerado pela ferramenta) que permite verificar o funcionamento do programa e das suas funcionalidades e restrições.

# Lista de requisitos e principais restrições

O objetivo do trabalho passa por modelar um sistema de indexação de publicações, ao estilo do *Academic Research*, da *Microsoft*. Para isso, foram identificados os seguintes requisitos e restrições do sistema:

**Requisitos:**

REQ1 – Adicionar autor (nome);

REQ2 – Adicionar afiliações (nome);

REQ3 – Relacionar um autor a uma afiliação;

REQ4 – Adicionar uma publicação (nome, descrição, autores, referências e a afiliação;

REQ5 – Adicionar interesse a um autor;

REQ6 – Calcular o caminho entre dois autores;

REQ7 – Calcular as citações pelo próprio autor;

REQ8 – Calcular as citações de um autor citado por outros autores;

REQ9 – Contar as publicações de um autor;

REQ10 – Calcular a distância entre dois autores;

REQ11 – Encontrar os autores por afiliação;

REQ12 – Listar as publicações por autor;

REQ13 – Calcular o gráfico de coautores para um autor.

**Principais restrições:**

RES1 – Uma publicação não pode fazer referência a uma outra, se a sua data for anterior à que pretende referir;

RES2 – 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;

RES3 – A data tem de ser válida;

RES4 – Ao inserir uma publicação o autor e as publicações referenciadas têm de pertencer ao indexador.

# Especificação das classes

**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

## Análise das restrições:

RES1 – Uma publicação não pode fazer referência a uma outra, se a sua data for anterior à que pretende referir:

...

pre compareDate(date, r.publication.date)

...

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;);

RES2 – 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:

...

for all ref in set references do

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

...

public addCitation : Reference ==> ()

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

pre compareDate(date, r.publication.date)

post r in set citations;

RES3 – A data tem de ser válida:

pre d <=31 and m <= 12 and

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

then d <= 30

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

RES4 – Ao inserir uma publicação o autor e as publicações referenciadas têm de pertencer ao indexador:

pre p not in set publications and p.authors subset dom authors and

forall ref in set p.references & ref.publication in set publications

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | REQ1 | REQ2 | REQ3 | REQ4 | REQ5 | REQ6 | REQ7 | REQ8 | REQ9 | REQ10 | REQ11 | REQ12 | REQ13 | RES1 | RES2 | RES3 | RES4 |
| TestIndexer.TestAddPublication | ✘ | ✘ |  | ✘ |  |  | ✘ | ✘ |  |  |  | ✘ |  | ✘ | ✘ | ✘ | ✘ |
| TestIndexer.TestCitations | ✘ | ✘ |  | ✘ |  |  |  |  |  |  |  |  |  | ✘ | ✘ | ✘ | ✘ |
| TestIndexer.TestDistanceBetween | ✘ | ✘ |  | ✘ |  |  |  |  |  | ✘ |  |  |  | ✘ | ✘ | ✘ | ✘ |
| TestIndexer.TestPathBetween | ✘ | ✘ |  | ✘ |  | ✘ |  |  |  |  |  |  |  | ✘ | ✘ | ✘ | ✘ |
| TestIndexer.TestCountPublicationsByAuthor | ✘ | ✘ |  | ✘ |  |  |  |  | ✘ |  |  |  |  | ✘ | ✘ | ✘ | ✘ |
| TestIndexer.TestCoAuthorPath | ✘ | ✘ |  | ✘ |  |  |  |  |  |  |  |  | ✘ | ✘ | ✘ | ✘ | ✘ |
| TestIndexer.TestAffiliations | ✘ | ✘ | ✘ |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| TestIndexer.TestGetAuthorsByAff | ✘ | ✘ | ✘ |  |  |  |  |  |  |  | ✘ |  |  |  |  |  |  |
| TestAuthor.TestInterests |  |  |  |  | ✘ |  |  |  |  |  |  |  |  |  |  |  |  |
| TestDate.GoodUsage |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | ✘ |  |
| TestPublication.TestCompareDate | ✘ | ✘ |  | ✘ |  |  |  |  |  |  |  |  |  | ✘ |  | ✘ |  |

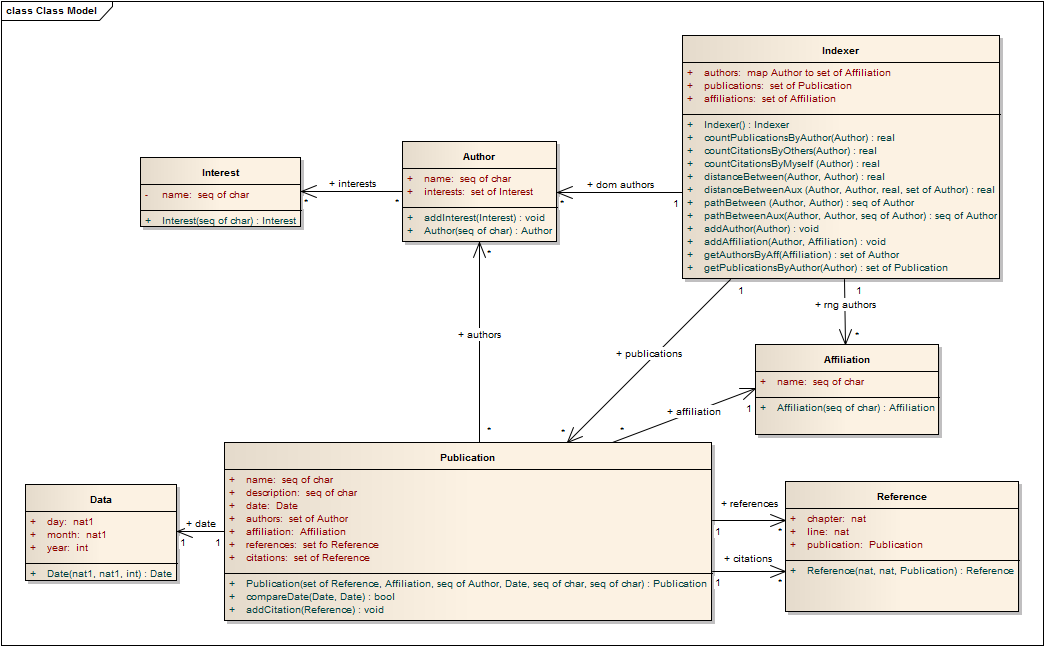
# Matriz de rastreabilidade dos testes com os requisitos e restrições

Só se considerou os testes da classe de testes *TestIndexer* e alguns testes da classe *TestAutor*, *TestDate* e *TestPublication*, pois estes acabam por englobar todos os testes das outras classes. Essas foram construídas ao longo da especificação e, por isso, apenas testam casos em particular, como por exemplo, criar um objecto do tipo Autor.

Tal como é possível verificar, os testes efetuados cobrem todos os requisitos e restrições definidos para o modelo.

# Diagrama conceptual de classes do sistema

O diagrama de classes é composto pelas classes definidas pelo enunciado do trabalho, as suas variáveis e as suas funções. A classe *Indexer* centraliza toda a informação e contém as funções mais importantes do sistema. O diagrama especifica também as relações entre as várias classes.



# Informação de cobertura dos testes

De seguida encontram-se as tabelas resultantes da execução do comando *pretty print* para todas as classes presentes no modelo.

|  |  |  |
| --- | --- | --- |
| ***name*** | ***#calls*** | ***coverage*** |
| Affiliation`Affiliation | 12 | 100% |
| ***total*** |  | ***100%*** |

|  |  |  |
| --- | --- | --- |
| ***name*** | ***#calls*** | ***coverage*** |
| Author`Author | 26 | 100% |
| Author`addInterest | 1 | 100% |
| ***total*** |  | ***100%*** |

|  |  |  |
| --- | --- | --- |
| ***name*** | ***#calls*** | ***coverage*** |
| Date`Date | 13 | 100% |
| ***total*** |  | ***100%*** |

|  |  |  |
| --- | --- | --- |
| ***name*** | ***#calls*** | ***coverage*** |
| Reference`Reference | 8 | 100% |
| ***total*** |  | ***100%*** |

|  |  |  |
| --- | --- | --- |
| ***name*** | ***#calls*** | ***coverage*** |
| Publication`Publication | 20 | 100% |
| Publication`addCitation | 4 | 100% |
| Publication`compareDate | 7 | 100% |
| ***total*** |  | ***100%*** |

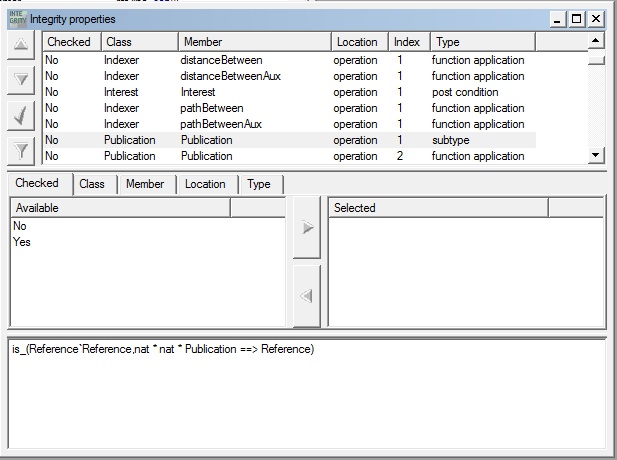
|  |  |  |
| --- | --- | --- |
| ***name*** | ***#calls*** | ***coverage*** |
| Interest`Interest | 2 | 100% |
| ***total*** |  | ***100%*** |

|  |  |  |
| --- | --- | --- |
| ***name*** | ***#calls*** | ***coverage*** |
| Indexer`Indexer | 8 | 100% |
| Indexer`addAuthor | 21 | 100% |
| Indexer`pathBetween | 1 | 100% |
| Indexer`coAuthorPath | 1 | 100% |
| Indexer`addAffiliation | 3 | 100% |
| Indexer`addPublication | 16 | 100% |
| Indexer`pathBetweenAux | 7 | 100% |
| Indexer`distanceBetween | 2 | 100% |
| Indexer`getAuthorsByAff | 1 | 100% |
| Indexer`distanceBetweenAux | 7 | 100% |
| Indexer`countCitationsByMyself | 1 | 100% |
| Indexer`countCitationsByOthers | 1 | 100% |
| Indexer`getPublicationsByAuthor | 1 | 100% |
| Indexer`countPublicationsByAuthor | 1 | 100% |
| ***total*** |  | ***100%*** |

Tal como é possível verificar, as classes de teste criadas permitiram testar integralmente o modelo, garantindo assim que todas as possíveis linhas de execução foram previstas e testadas de acordo, sendo o resultado o esperado.

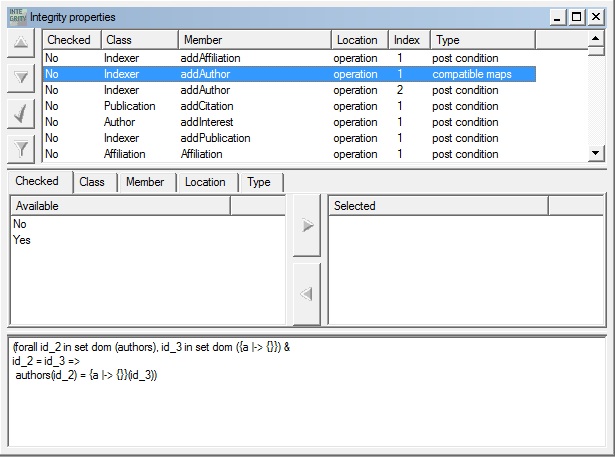
# Análise da consistência do modelo

Recorrendo à ferramenta que permite testar a integridade do modelo, foram analisados os casos em que realmente há alguma sugestão para melhorar a sua consistência.



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

Como podemos verificar a sugestão apresentada não é necessária, visto que ao criar um novo objeto da classe References garantimos que este é dessa classe.



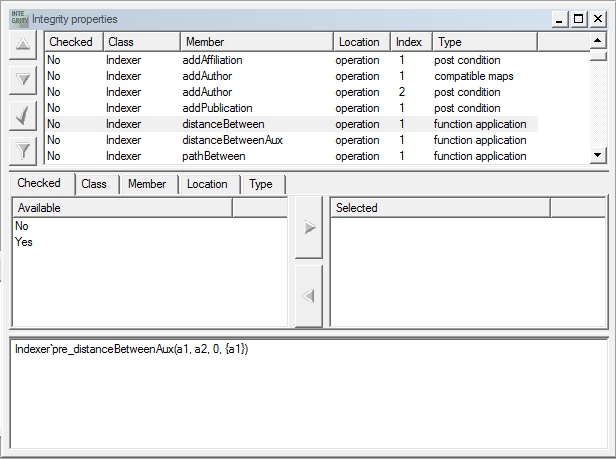
public addAuthor: Author ==> ()

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

pre a not in set dom authors

post a in set dom authors;

A sugestão indica que é necessário verificar algo caso os autores sejam iguais. Como é possível verificar, a pré-condição garante à partida que não estamos a adicionar um autor repetido e, como tal, a verificação torna-se desnecessária.



Ao longo da análise feita pela ferramenta, verificamos várias incidências de situações como a anterior. Isto acontece sempre que dentro de uma função invocamos uma outra função dessa mesma classe. É sugerido verificar que a função chamada pertence à classe em questão mas visto que a função se encontra criada não é necessário ter estes casos em conta.

Todas as restantes sugestões, resultantes do teste de integridade usando a ferramenta VDM++ Toolbox, podemos considerar falso-positivos visto que são casos em que a pré ou pós condição sugerida já é verificada e a integridade é assegurada apesar de a ferramenta não o detetar.

# Estimativa do tempo gasto

* Especificação – 15 horas
* Construção de casos de teste – 10 horas
* Implementação interface com o código gerado – 5 horas

# Conclusão

Tal como era proposto inicialmente este trabalho permitiu tomar conhecimento de um tipo diferente de desenvolvimento daquele que é abordado ao longo do curso. A análise formal do problema permite perceber quais as restrições que o programa deve cumprir e usando o VDM++ conseguimos especificar essas restrições para garantir a qualidade do produto.

A especificação de invariantes, pré e pós condições permite garantir que tais restrições não são relegadas para segundo plano, e eventualmente esquecidas, numa fase posterior do desenvolvimento, garantindo assim a integridade do produto criado com um elevado nível de confiança.