Um estudo de caso usando o método formal Z para especificação de um sistema

Bruno Biribio Woerner

Orientado por José Roque Voltolini da Silva

Roteiro da apresentação

- Introdução
- Objetivos
- Fundamentação teórica
- Desenvolvimento
- Conclusão
- Extensões

Introdução

- Problemas no desenvolvimento de software
- Solução proposta

Objetivos

- Implementar um sistema utilizando o método formal Z
- Desenvolver um sistema de biblioteca
- Realizar a implementação a partir da especificação
- Construir um roteiro desde a especificação até a implementação em Java

Fundamentação teórica

- Processo de desenvolvimento
- Especificação formal
- Notação Z
- Mapeamento da notação Z para Java
- Ambiente de desenvolvimento Z/EVES
- Trabalhos correlatos

Processo de desenvolvimento

- Ciclo de vida do software: conceber uma idéia, seguir o desenvolvimento do produto, realizar as vendas, atingir sua maturidade e podendo futuramente torna-se obsoleto
- Desenvolvimento do software: análise de requisitos, especificação do sistema, projeto ou *design,* implementação, testes e integração, suporte e otimizações.

Processo de desenvolvimento

- Problemas relatados na utilização de requisitos: imprecisão, contradição, imperfeito, requisitos misturados, ambigüidade e diferentes níveis de abstração

Especificação Formal

- Surgiu da busca da necessidade existente na computação em desenvolver softwares precisos e concisos; de fácil manipulação e entendimento; algo que se adapte ao processo de desenvolvimento de software
- Algumas áreas já possuem este mecanismo incorporado, como é o exemplo da arquitetura e engenharia

Especificação Formal

- Na área de computação, os métodos matemáticos utilizados, a matemática discreta e a lógica matemática, estavam em seu estado inicial no início do século XX.
- Aplicações diretas do uso destes métodos não chegam a 50 anos.

Especificação Formal

- boa abstração e clareza do comportamento do software
- maior interação com o cliente ainda na fase de especificação
- compreensão do sistema como um todo torna-se mais fácil

Notação Z

- Um dos maiores casos de sucessos da utilização de Z foi a realizada no projeto *Customer Information Control System* (CICS) da IBM
- Mostrou-se de fácil aprendizado e simples uso, mesmo para programadores sem prévia experiência com matemática, além de aumentar a qualidade e confiabilidade do código produzido

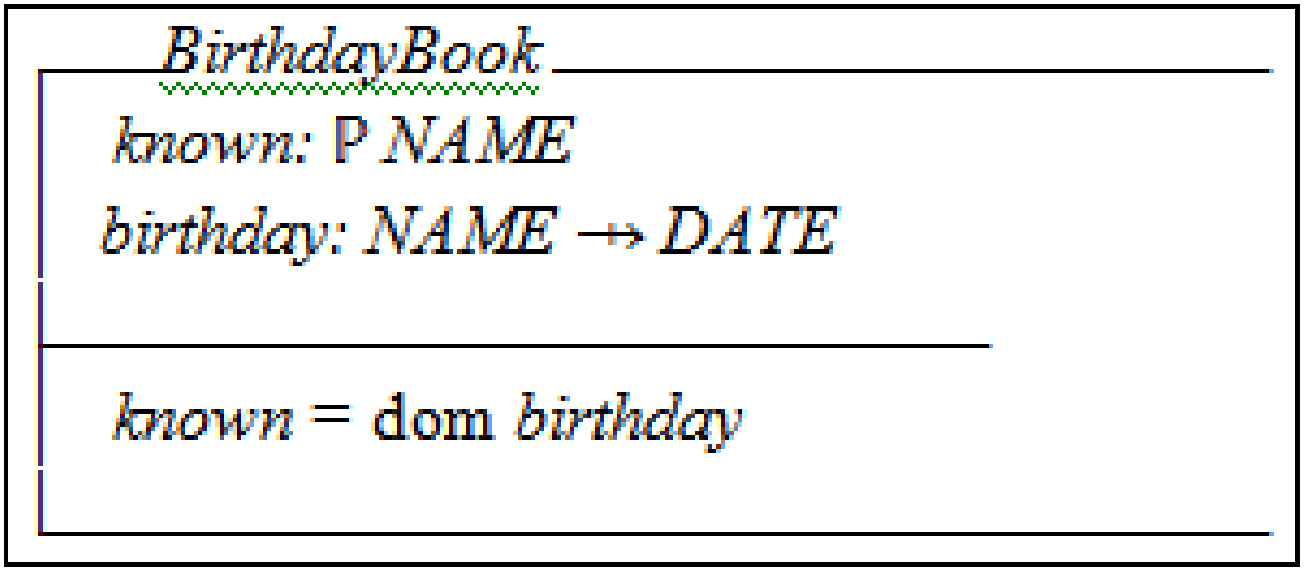
Notação Z

- A partir do sucesso obtido, em abril de 1992, o *Queen's Award for Technological Achievement* foi entregue a IBM e ao Laboratório de Computação de Oxford, segundo o título “desenvolvimento e uso de método de programação avançada que reduz o custo de desenvolvimento e aumenta significativamente a qualidade e confiabilidade”

Notação Z

- sistemas não tolerante a falhas, como: sinalização férrea, aparelhos médicos, sistemas de energia nuclear
- sistemas seguros, como: sistemas de transação bancária, comunicação
- sistemas em geral, como: linguagem de programação, processadores de ponto flutuante
- formalização de semânticas de outras linguagens, especialmente em documentos de padrões

Notação Z



- Parte declarativa (acima da linha central)
- Parte predicativa(abaixo da linha central)

Notação Z

$$known = \{John, Mike, Susan\}$$

$$birthday = \{John \mapsto 25\text{-}Mar, Mike \mapsto 20\text{-}Dec, Susan \mapsto 20\text{-}Dec\}$$

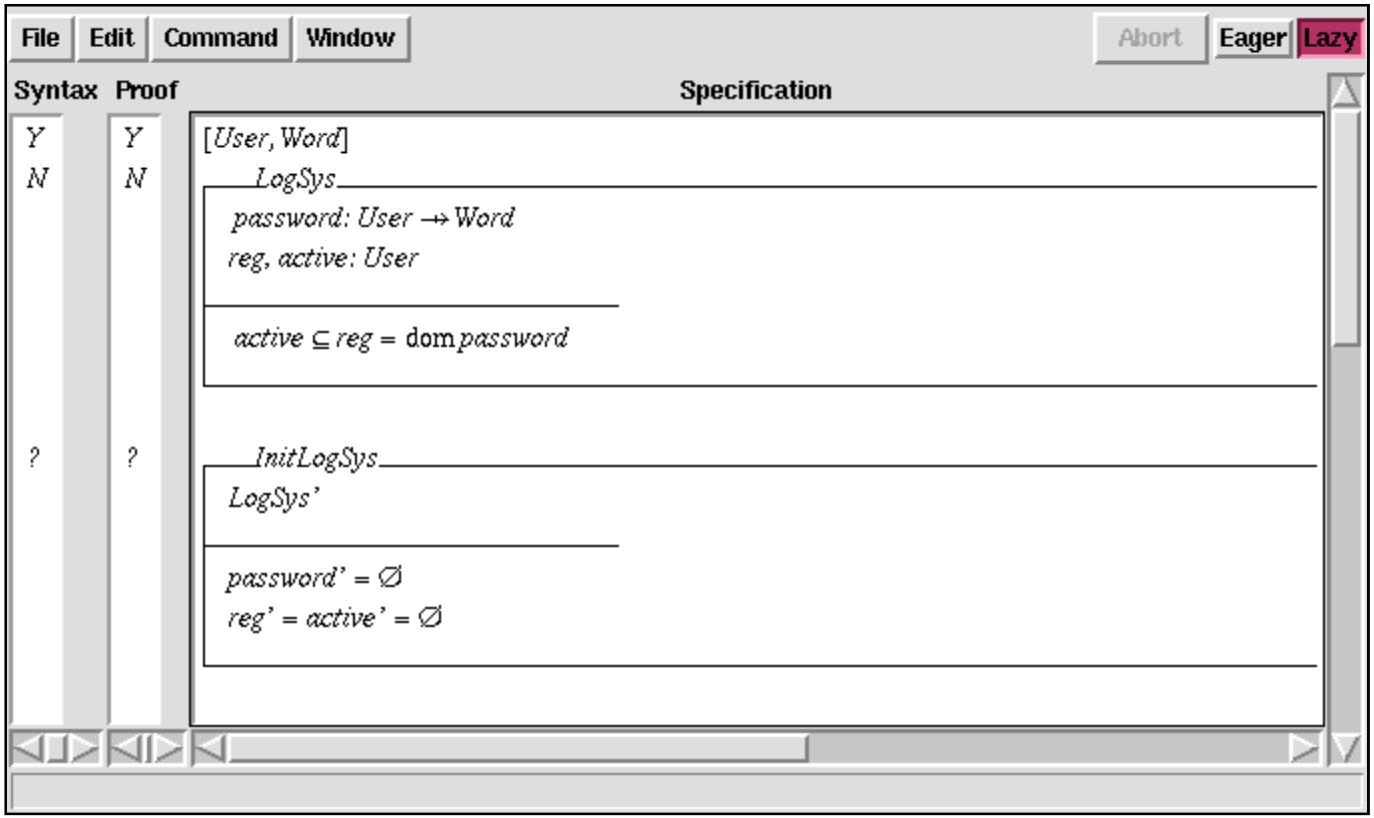
Mapeamento da notação Z para Java

- Não há conhecimento de ferramentas maduras para a geração de código a partir de Z para Java
- Ferramenta JZed-Gen

Ambiente de desenvolvimento Z/EVES

- Ferramenta para analisar especificações em Z
- Também realiza *parse*, checagem de tipo, checagem de domínios, expansão de *schemas*, cálculo de pré-condições, refinamento de provas e prova de teoremas.

Ambiente de desenvolvimento Z/EVES



Trabalhos correlatos

- Sistema de Iluminação da família do Airbus A320/330 (Hammer e Peleska)
- Sistema de compras de CD utilizando a notação formal Método de desenvolvimento Viena (VDM) (Findeiss)
- Framework de testes. Geração de especificação de testes a partir de uma especificação em Z (Stocks)

Desenvolvimento

- Levantamento de requisitos
- Especificação do sistema
- Implementação Java a partir da especificação em Z
- Especificação da interface
- Estudo de caso

Desenvolvimento

Levantamento de requisitos

- Requisitos Funcionais
  - inclusão, alteração e remoção de leitores da biblioteca
  - inclusão, alteração e remoção das diversas categorias de leitores
  - inclusão, alteração e remoção das diversas categorias de obras literárias
  - a inclusão, alteração e remoção das obras literárias da biblioteca

Desenvolvimento

Levantamento de requisitos

- Requisitos Funcionais
  - inclusão, alteração e remoção de funcionários da biblioteca
  - processamento da reserva da obra literária
  - empréstimo de obras literária para um leitor
  - permitir a devolução de uma obra literária por um leitor

Desenvolvimento

Levantamento de requisitos

- Requisitos Funcionais
  - permitir a geração de um relatório com as obras que estão em empréstimo
  - permitir a geração de um relatório com as obras que estão em atraso
  - permitir a consulta da situação de uma obra literária

Desenvolvimento

Levantamento de requisitos

- Requisitos Não-Funcionais
  - especificar o software utilizando a notação formal Z
  - implementar testes utilizando a ferramenta JUnit
  - utilizar o ambiente de desenvolvimento Eclipse

Desenvolvimento
Especificação

- *Schemas* definidos em Z
- Interfaces construídas sem utilização de notação formal

Desenvolvimento

Especificação de Date

Date

$day: 1 .. 31$

$month: 1 .. 12$

$year: 1900 .. 9999$

$value: \mathbb{N}$

$month \notin \{2, 4, 6, 9, 11\}$

$\lor\ month \in \{4, 6, 9, 11\} \land day \leqslant 30$

$\lor\ month = 2 \land year \notin leapYear \land day \geqslant 28$

$\lor\ month = 2 \land year \in leapYear \land day \geqslant 29$

$value = value + (year - 1900) * 366 \Leftrightarrow year \in leapYear$

$value = value + (year - 1900) * 365 \Leftrightarrow year \notin leapYear$

$value = value + month * 30 \Leftrightarrow month \in \{4, 6, 9, 11\}$

$value = value + month * 31 \Leftrightarrow month \notin \{2, 4, 6, 9, 11\}$

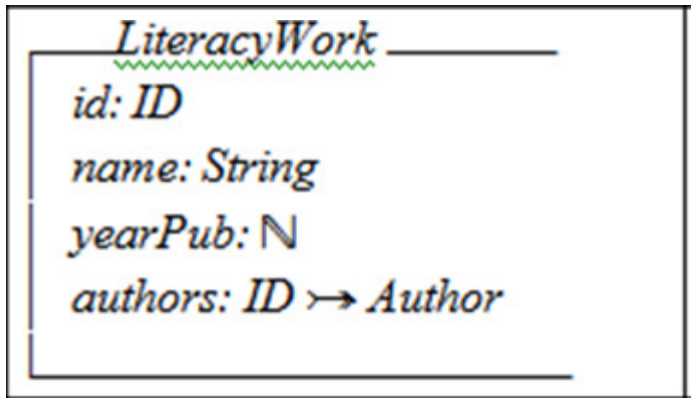
$value = value + month * 29 \Leftrightarrow month = 2 \land year \in leapYear$

$value = value + month * 28 \Leftrightarrow month = 2 \land year \notin leapYear$

$value = value + day$

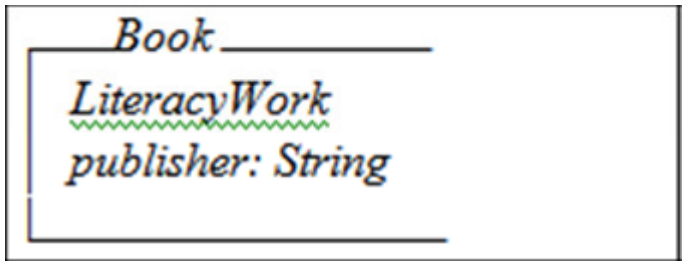
Desenvolvimento

Especificação de LiteracyWork



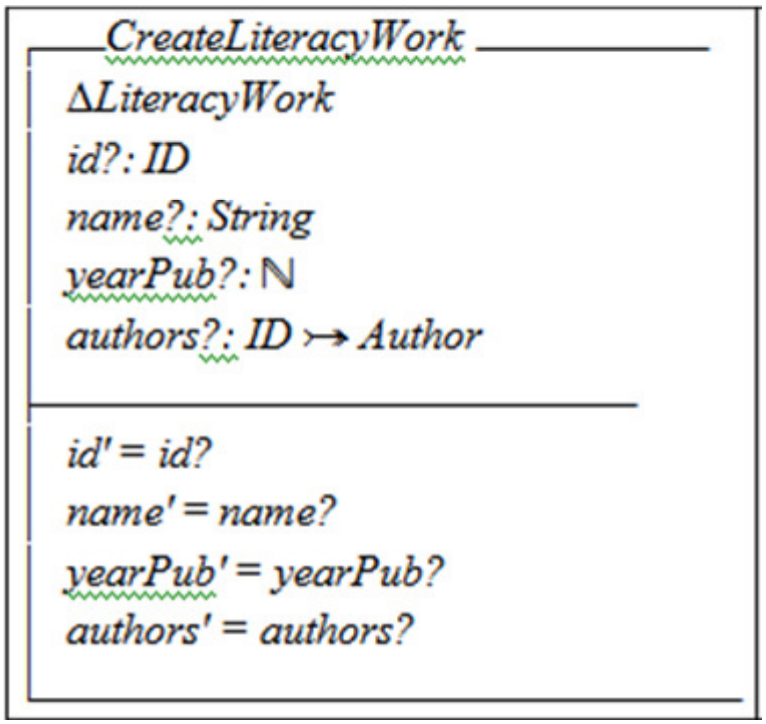
Desenvolvimento

Especificação de Book



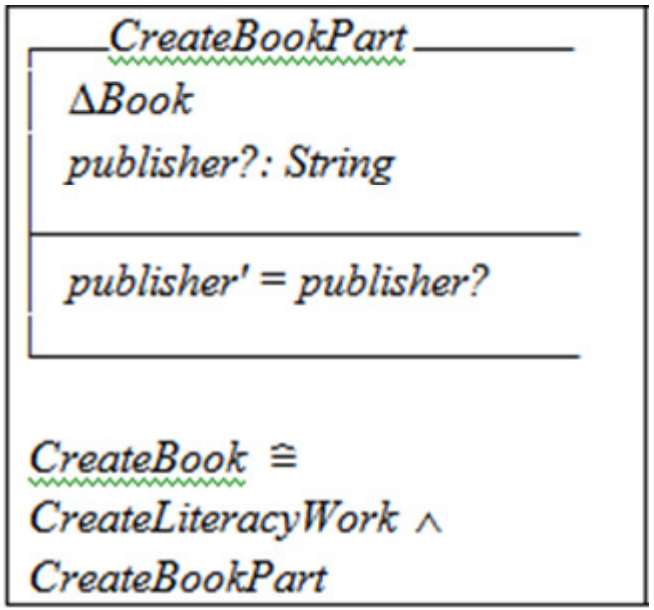
Desenvolvimento

Especificação de CreateLiteracyWork



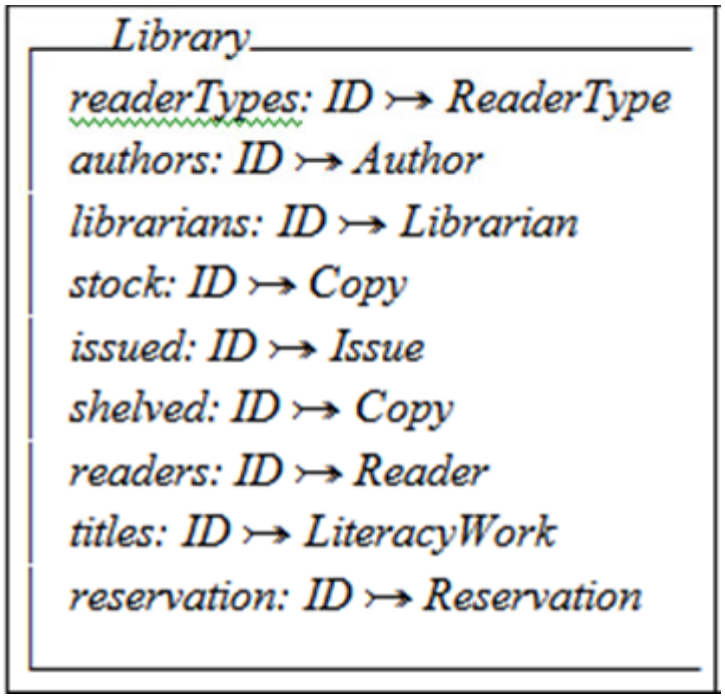
Desenvolvimento

Especificação de CreateBookPart



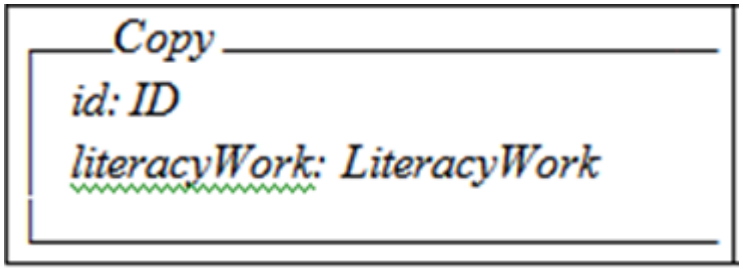
Desenvolvimento

Especificação de Library



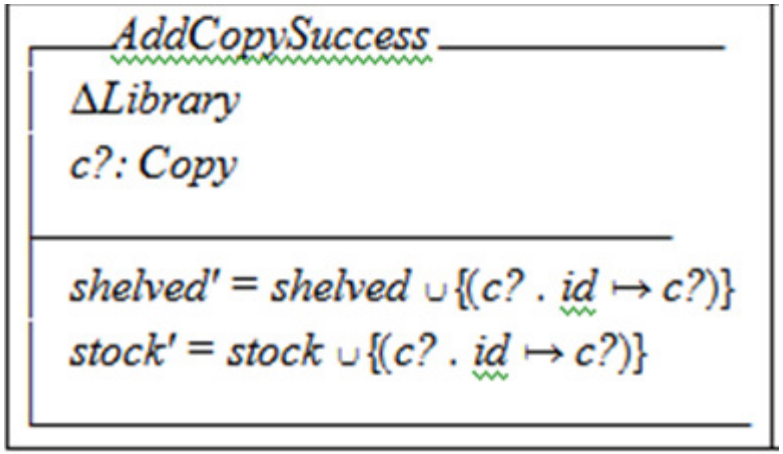
Desenvolvimento

Especificação de Copy



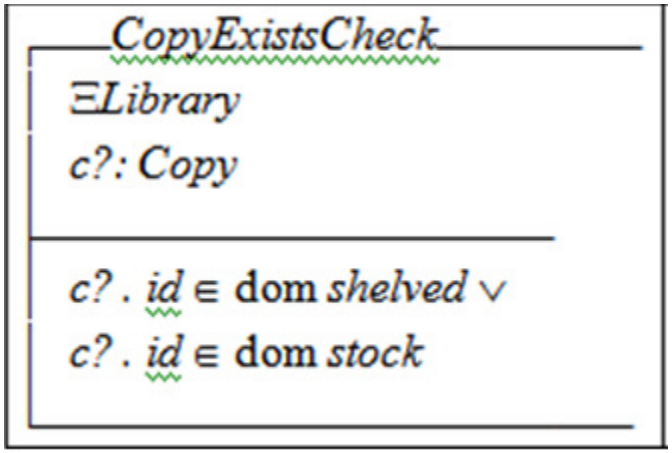
Desenvolvimento

Especificação de AddCopySuccess



Desenvolvimento

Especificação de CopyExistsCheck



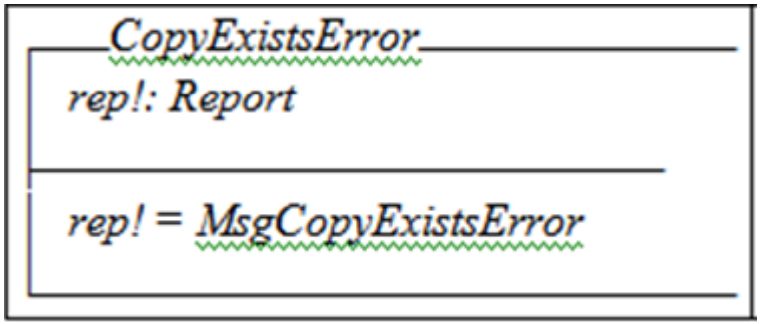
Desenvolvimento

Especificação de CopyNotExistsCheck

$$CopyNotExistsCheck \triangleq \neg CopyExistsCheck$$

Desenvolvimento

Especificação de CopyExistsError



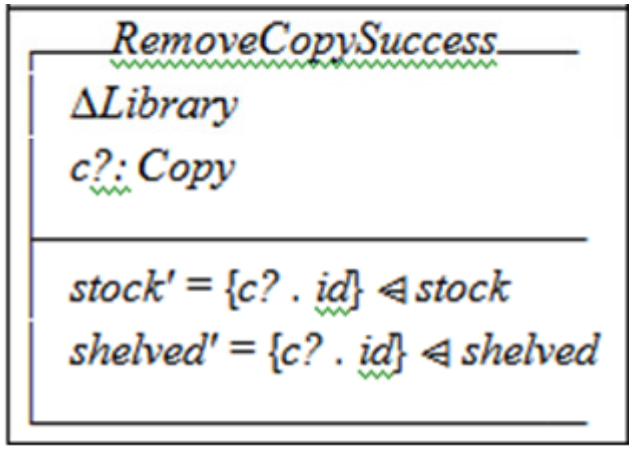
Desenvolvimento

Especificação de AddCopy

$$
\begin{aligned}
&AddCopy \triangleq \\
&\quad CopyNotExistsCheck \land \\
&AddCopySuccess \lor \\
&CopyExistsCheck \land \\
&CopyExistsError
\end{aligned}
$$

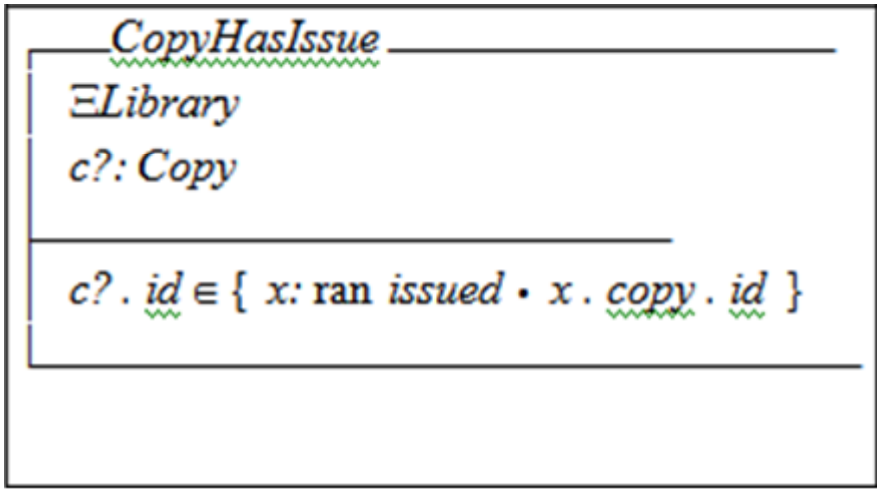
Desenvolvimento

Especificação de RemoveCopySuccess



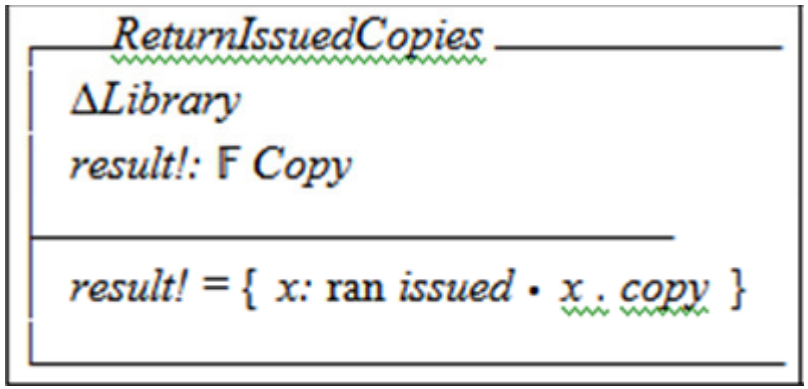
Desenvolvimento

Especificação de CopyHasIssue



Desenvolvimento

Especificação de ReturnIssuedCopies



Implementação

Mapeamento Z -> Java

- Mapeamento das estruturas definidas em Z para Java

Implementação

Mapeamento Z -> Java

Z	Java
LiteracyWork *id: ID* *name: String* *yearPub: ℕ* *authors: ID ⤖ Author*	`public class LiteracyWork {` `  public ID id;` `  public String name;` `  public int yearPub;` `  public ZRelation<ID, Author> authors;` `}`

Implementação

Mapeamento Z -> Java

Z	Java
Book *LiteracyWork* *publisher: String*	`public class Book extends LiteracyWork {` `  public String publisher;` `}`

Implementação

Mapeamento Z -> Java

Z	Java
CreateLiteracyWork $\Delta LiteracyWork$ $id?: ID$ $name?: String$ $yearPub?: \mathbb{N}$ $authors?: ID \rightarrowtail Author$ --- $id' = id?$ $name' = name?$ $yearPub' = yearPub?$ $authors' = authors?$	`public class LiteracyWork {` `  public ID id;` `  public String name;` `  public int yearPub;` `  public ZRelation<ID, Author> authors;` `  public LiteracyWork(ID id, String name, int yearPub, ZRelation<ID, Author> authors) {` `  this.id = id;` `  this.name = name;` `  this.yearPub = yearPub;` `  this.authors = authors;` `  }` `}`

Implementação

Mapeamento Z -> Java

Z	Java
CreateBookPart $\Delta Book$ $publisher?: String$ --- $publisher' = publisher?$ *CreateBook* $\widehat{=}$ *CreateLiteracyWork* $\wedge$ *CreateBookPart*	`public class Book extends LiteracyWork {` `  public String publisher;` `  public Book(ID id, String name, int yearPub,` `ZRelation<ID, Author> authors, String publisher) {` `    super(id, name, yearPub, authors);` `    this.publisher = publisher;` `  }` `}`

Implementação

Mapeamento Z -> Java

Z	Java
Library *readerTypes: ID ⤖ ReaderType* *authors: ID ⤖ Author* *librarians: ID ⤖ Librarian* *stock: ID ⤖ Copy* *issued: ID ⤖ Issue* *shelved: ID ⤖ Copy* *readers: ID ⤖ Reader* *titles: ID ⤖ LiteracyWork* *reservation: ID ⤖ Reservation*	`public class Library {` `public ZRelation<ID, Librarian> librarians;` `public ZRelation<ID, ReaderType> readersTypes;` `public ZRelation<ID, Author> authors;` `public ZRelation<ID, Copy> stock;` `public ZRelation<ID, Issue> issued;` `public ZRelation<ID, Copy> shelved;` `public ZRelation<ID, Reader> readers;` `public ZRelation<ID, LiteracyWork> titles;` `public ZRelation<ID, Issue> overdue;` `public ZRelation<ID, Reservation> reserved;` `...` `}`

Implementação

Mapeamento Z -> Java

Z	Java
Copy *id: ID* *literacyWork: LiteracyWork*	`public class Copy {` `  public ID id;` `  public LiteracyWork literacyWork;` `}`

Implementação

Mapeamento Z -> Java

Z	Java
AddCopySuccess $\Delta Library$ $c?: Copy$ --- $shelved' = shelved \cup \{(c? \,.\, id \mapsto c?)\}$ $stock' = stock \cup \{(c? \,.\, id \mapsto c?)\}$	`public class Library {` `  ...` `  private void addCopySuccess(Copy c) {` `  shelved.put(c.id, c);` `  stock.put(c.id, c);` `  }` `}`

Implementação

Mapeamento Z -> Java

Z	Java
CopyExistsCheck $\Xi Library$ $c? : Copy$ --- $c?.id \in \text{dom } shelved \lor$ $c?.id \in \text{dom } stock$	`public class Library {` `  ...` `  private boolean copyExistsCheck(Copy c) {` `  return shelved.domain().contains(c.id) ||` `  stock.domain().contains(c.id);` `  }` `}`

Implementação

Mapeamento Z -> Java

Z	Java
$CopyNotExistsCheck \triangleq \neg$ $CopyExistsCheck$	`public class Library {` `  ...` `  private boolean copyNotExistsCheck(Copy c) {` `  return !copyExistsCheck(c);` `  }` `}`

Implementação

Mapeamento Z -> Java

Z	Java
CopyExistsError *rep!: Report* --- *rep! = MsgCopyExistsError*	public class Library { ... private void copyExistsError() { throw new ReportException(Report.CopyAlreadyExists); } }

Implementação

Mapeamento Z -> Java

Z	Java
AddCopy ≙ *CopyNotExistsCheck* ∧ *AddCopySuccess* ∨ *CopyExistsCheck* ∧ *CopyExistsError*	`public class Library {` `  ...` `  public void addCopy(Copy c) {` `  if (copyNotExistsCheck(c)) {` `  addCopySuccess(c);` `  } else if (copyExistsCheck(c)) {` `  copyExistsError();` `  } else {` `  throw new IllegalStateException();` `  }` `  }` `}`

Implementação

Mapeamento Z -> Java

Z	Java
RemoveCopySuccess $\Delta Library$ $c? : Copy$ --- $stock' = \{c?.id\} \lhd stock$ $shelved' = \{c?.id\} \lhd shelved$	`public class Library {` `  ...` `  public void removeCopy(Copy c) {` `  stock = stock.domain_restriction(stock, c.id);` `  shelved = shelved.domain_restriction(shelved, c.id);` `  }` `}`

Implementação

Mapeamento Z -> Java

Z	Java
CopyHasIssue $\exists Library$ $c? : Copy$ --- $c?.id \in \{ x : \text{ran } issued \bullet x.copy.id \}$	`public class Library {` `  ...` `  public boolean copyHasIssue(Copy c) {` `  ZSet<Issue> issues = issued.range();` `  for (Issue issue : issues) {` `  if (issue.copy.id.equals(c.id)) {` `  return true;` `  }` `  }` `  return false;` `  }` `}`

Implementação

Mapeamento Z -> Java

Z	Java
ReturnIssuedCopies $\Delta Library$ $result! : \mathbb{F}\ Copy$ $result! = \{\ x : \mathrm{ran}\ issued \bullet x.copy\ \}$	`public ZSet<Copy> returnIssuedCopies() {` `  ZSet<Issue> issued = library.issued.range();` `  ZSet<Copy> copies = new BasicSet<Copy>();` `  for (Issue issue : issued) {` `  Copy copy = issue.copy;` `  copies.add(copy);` `  }` `  return copies;` `}`

Implementação

Prova de propriedades

- Não foi realizado um extenso estudo de prova de propriedades neste trabalho
- Pode ser trabalhoso caso não haja domínio da notação Z

Implementação

Prova de propriedades

$$AddLibrarian \mathrel{\widehat=} LibrarianNotExistsCheck \land AddLibrarianSuccess \lor LibrarianExistsCheck \land LibrarianExistsError$$

LibrarianExistsCheck

$$\Xi Library$$
$$l? : Librarian$$
$$l?.id \in \mathrm{dom}\ librarians$$

LibrarianExistsError

$$rep! : Report$$
$$rep! = MsgLibrarianExistsError$$

Implementação

Prova de propriedades

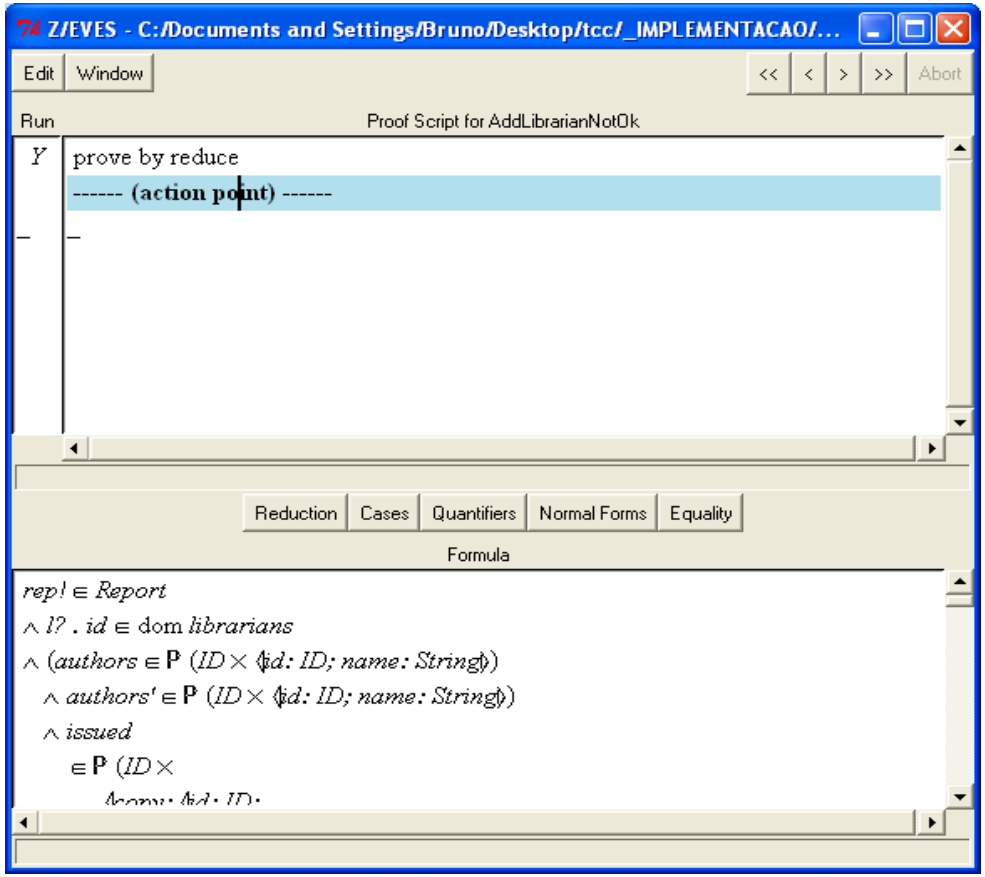
theorem *AddLibrarianNotOk*

$$\forall AddLibrarian \mid l? \,.\, id \in \mathrm{dom}\ librarians \bullet rep! = MsgLibrarianExistsError$$

$$rep! \in Report$$
$$\wedge\ l? \,.\, id \in \mathrm{dom}\ librarians$$
$$\ldots$$
$$\wedge\ rep! = MsgLibrarianExistsError$$
$$\ldots$$
$$\Rightarrow rep! = MsgLibrarianExistsError$$

Implementação

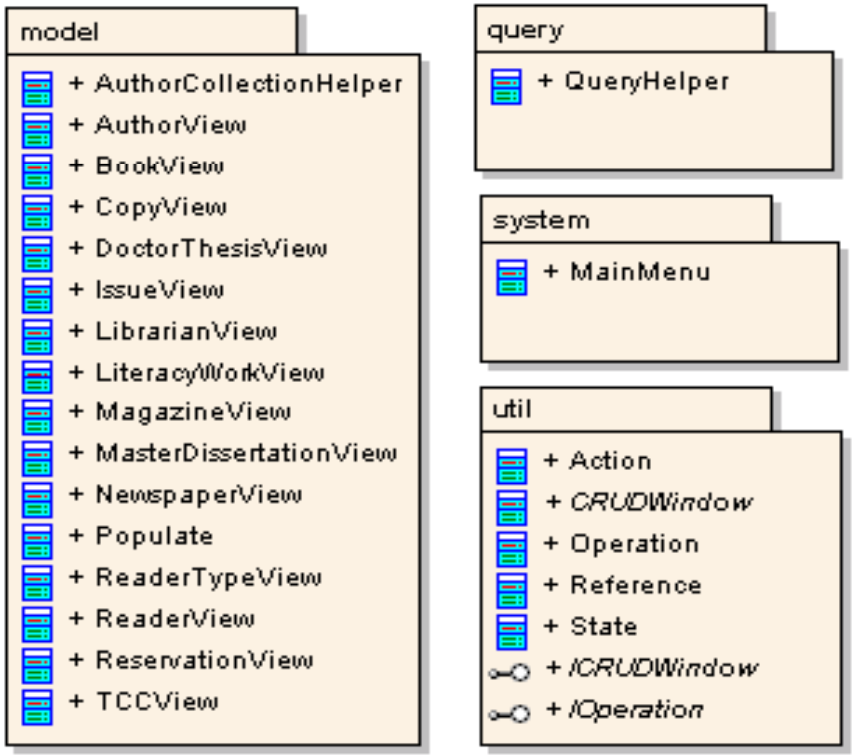
Prova de propriedades



Implementação
Interface

- Metodologia “tradicional”
- Utilização do conceito de orientação a objetos

Implementação
Interface

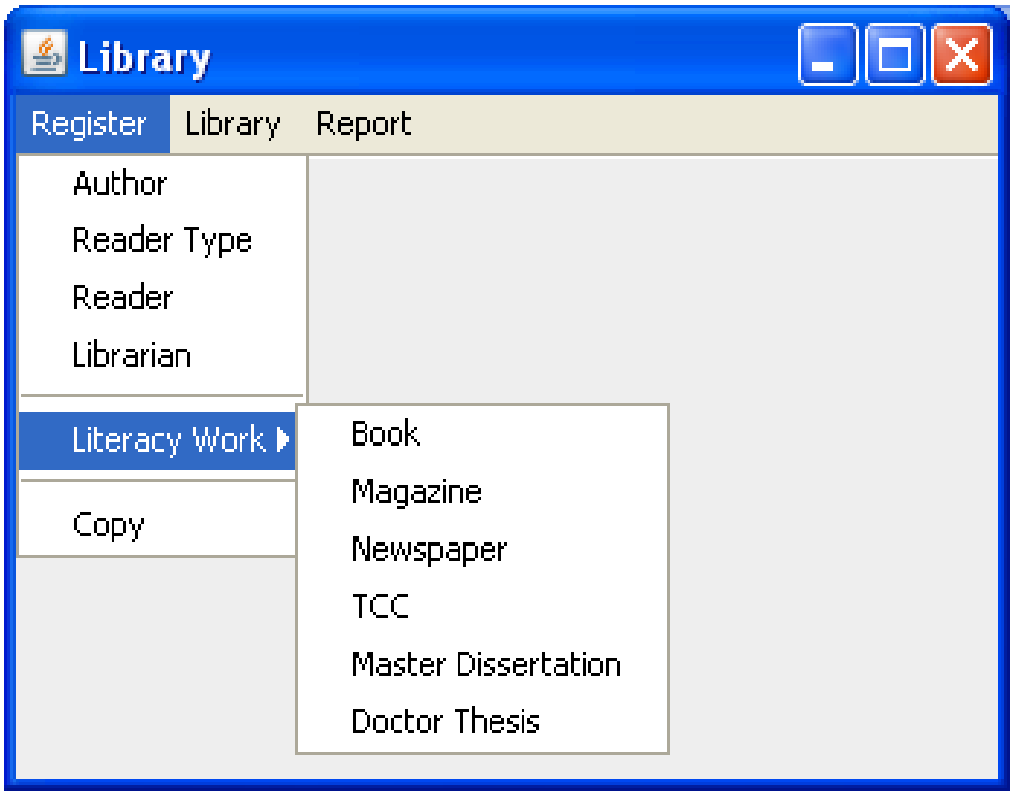


Implementação
Estudo de caso

- Implementação de um sistema de biblioteca

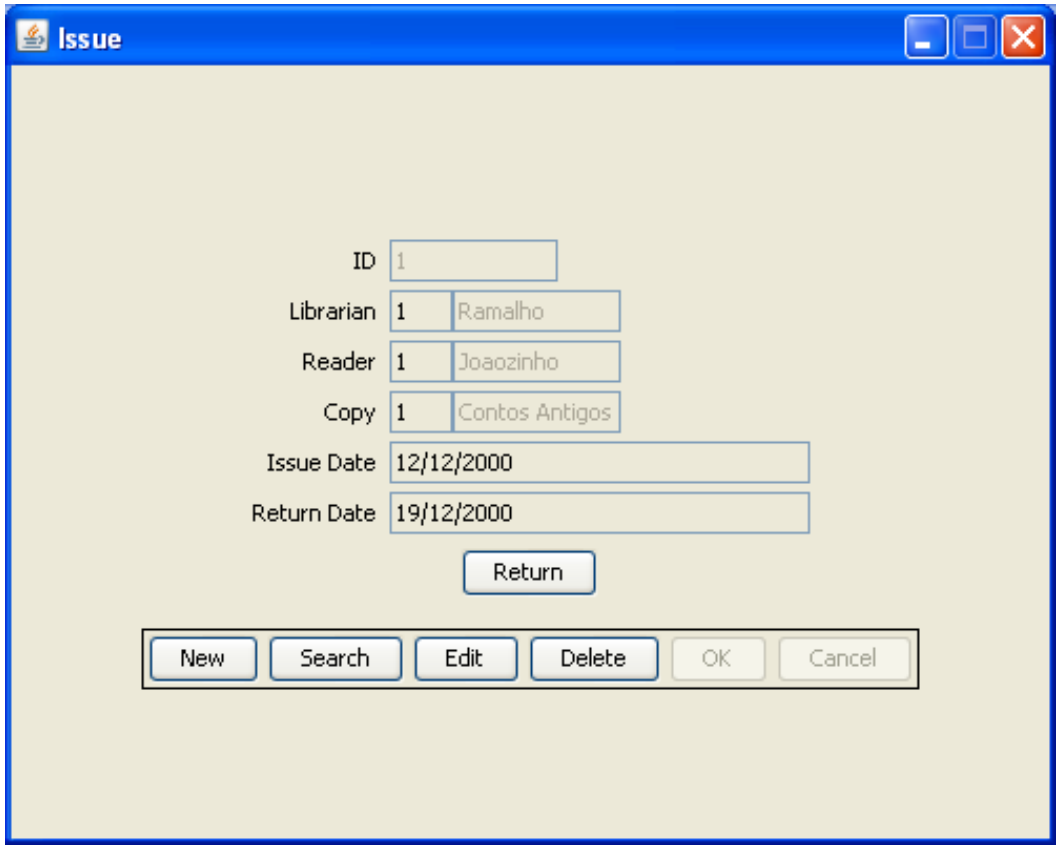
Implementação

Estudo de caso



Implementação

Estudo de caso



Resultados e discussão

- Não é a panacéia do desenvolvimento do software
- Não é necessário ser um matemático para aplicar os métodos formais
- A utilização da notação Z tornou muito mais fácil a fase de implementação
- Produtividade na realização de testes utilizando a notação formal
- Possibilita melhor confrontamento com os requisitos

Conclusão

- Principais objetivos alcançados com sucesso
- Maior facilidade para contemplar requisitos na implementação
- Falta de ferramentas para trabalhar com a notação Z
- Falta de processo para desenvolver métodos formais
- Utilizar métodos formais é útil e totalmente viável no desenvolvimento de software

Extensões

- Implementar uma ferramenta que possa converter especificações em Z para código Java
- Implementar uma ferramenta para realizar a construção de especificações em Z
- Implementar um ferramenta que auxilie na geração de testes a partir da especificação
- Implementar a especificação de prova utilizando a especificação do sistema construído

Extensões

- Implementar o refinamento da especificação
- Especificar em Z as interfaces do sistema de biblioteca
- Focar a especificação de um sistema para um ambiente específico
- Construir uma metodologia para o desenvolvimento de software utilizando especificação formal