

Anexo 1

Especificação de Requisitos de Software

1.Requisitos de software

Actualmente, a tecnologia desenvolvida pela SISCOG dispõe de um conjunto de interfaces de programação que são disponibilizadas aos técnicos utilizadores dessa mesma tecnologia. Tais interfaces têm documentação associada e que pode ser consultada em ficheiros de código fonte Lisp sob a forma de documentação interna. Por forma a facilitar a consulta dessa informação pretende-se que o «Gerador de Manuais de Referência» (adiante designado por *sistema*) proceda à leitura desses ficheiros de código fonte, processe *somente* a documentação interna relevante, e por fim a exporte para um formato de documentação de fácil consulta e leitura.

A forma e o conteúdo da documentação interna está sujeita a um conjunto de regras que deverão guiar a análise feita pelo sistema. Desta forma, a execução do sistema permitirá também validar as regras a que essa documentação interna deve obedecer, informando o utilizador da existência ou não de alguma violação.

Pretende-se que o sistema realize duas tarefas distintas, a saber:

1. Extracção de dados
 - Leitura de texto em código fonte
 - Processamento de documentação interna
 - Escrita dos dados processados
2. Exportação de dados

- Leitura de dados processados
- Geração de documentação numa dada linguagem¹, com base nos dados processados
- Escrita da documentação gerada

O resultado de uma extracção de dados deve poder ser guardado para referência futura. Desta forma será possível fazer mais do que uma exportação a partir de um qualquer conjunto de dados extraído previamente. A Figura 1 ilustra a forma como as duas fases se interligam.

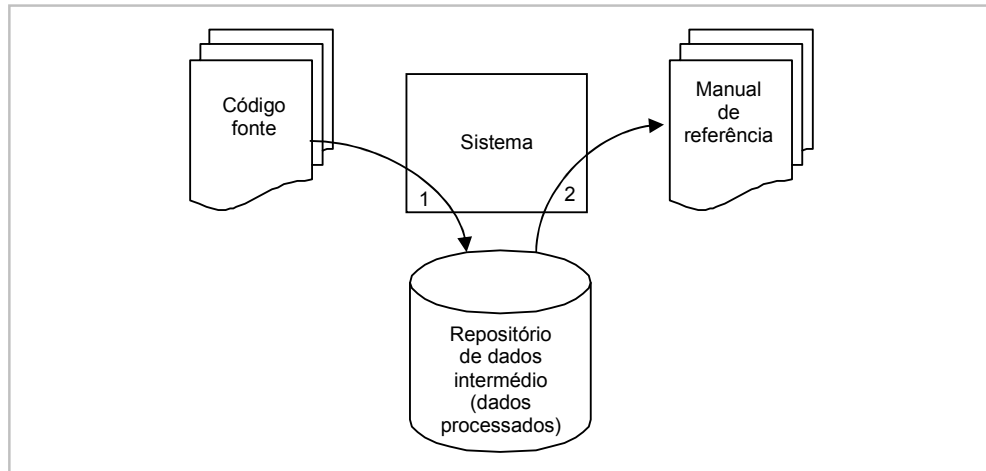


Figura 1: Interligação das principais fases de execução do sistema.

Seguidamente são apresentados os requisitos de *software* que o sistema deve cumprir. Estão divididos em duas secções que correspondem às duas fases acima referidas.

¹ Por exemplo, em HTML.

1.1. Extracção de dados

Esta secção contém as especificações referentes à leitura e interpretação da documentação interna existente em ficheiros de código fonte e ao armazenamento, de forma persistente, do conjunto de dados resultante.

1.1.1. Dados de entrada

Os ficheiros de código fonte na linguagem de programação Lisp estão localizados num sistema de ficheiros.

✓ *Requisito de software 01*

O sistema deve ler um ou mais ficheiros, de um ou mais tipos, contidos numa directoria de um sistema de ficheiros, e nas suas subdirectorias, se existirem.

✓ *Requisito de software 02*

Deve ser possível indicar ao sistema qual o caminho da directoria de topo que contém os ficheiros a serem processados.

✓ *Requisito de software 03*

Deve ser possível indicar ao sistema quais os tipos de ficheiros que devem ser processados.

1.1.2. Processamento

A documentação interna contida nos ficheiros de código fonte está sujeita a regras de forma e conteúdo definidas pela SISCOG.

✓ *Requisito de software 04*

Deve ser possível indicar ao sistema quais as regras que regem a forma e o conteúdo da documentação interna.

✓ *Requisito de software 05*

O sistema deve ser capaz de assinalar um erro sempre que a documentação interna nos ficheiros de código fonte não obedeça às regras em utilização. A informação

sobre o erro deve incluir o nome e localização do ficheiro e a posição, dentro do ficheiro, onde o erro foi encontrado.

1.1.3. Dados de saída

Por forma a poder exportar a informação extraída dos ficheiros de código fonte para um ou mais formatos de documentação, em diferentes momentos no tempo, pretende-se que sistema seja capaz de guardar essa informação num formato de dados intermédio.

✓ *Requisito de software 06*

O sistema deve gerar um ficheiro com uma representação de dados intermédia obtida a partir da informação extraída dos ficheiros de código fonte.

1.2. Exportação de dados

Esta secção contém as especificações referentes ao carregamento de dados armazenados no repositório intermédio e à consequente geração do manual de referência.

1.2.1. Dados de entrada

Os dados de entrada a que esta secção se refere são os dados que foram recolhidos pelo sistema no processo de importação.

✓ *Requisito de software 09*

Deve ser possível indicar ao sistema qual o ficheiro de dados intermédio a partir do qual se pretende exportar informação para um formato de documentação.

1.2.2. Dados de saída

Os dados de saída a que esta secção se refere são os dados exportados para documentos destinados a serem lidos pelo utilizador.

✓ *Requisito de software 07*

O sistema deve suportar pelo menos um formato de documentação para o qual se possa exportar a informação contida na representação de dados intermédia.

✓ *Requisito de software 08*

Deve ser possível indicar ao sistema qual o formato de documentação que deve ser usado na exportação da informação contida no ficheiro de dados intermédio.

✓ *Requisito de software 10*

Deve ser possível indicar ao sistema qual a informação a ser exportada a partir de um ficheiro de dados intermédio.

1.3. Regras

1.3.1. Referências

Internal Documentation Rules, WI.DMP.03

1.3.2. Identificação do texto a ser processado

Na SISCOG, a documentação interna compreende principalmente duas partes: (1) o cabeçalho do ficheiro, no início de cada *Ficheiro de Código Fonte*; (2) os cabeçalhos referentes a cada *Elemento de Código Fonte*.

Os segundos contêm parte da informação que caracteriza o elemento, sob a forma de atributos do elemento. A restante informação é associada também a atributos, mas estes são obtidos de forma indirecta, quer através de dados localizados noutros pontos do mesmo ficheiro de código fonte ou em outros ficheiros de código fonte, quer a partir de atributos de outros elementos de código fonte. Designamos cada um destes atributos, directos ou indirectos, por *Atributo de Elemento de Código Fonte*.

Os primeiros contêm também informação que caracteriza o ficheiro, a qual é associada a atributos desse ficheiro. Designamos cada um destes atributos por *Atributo de Ficheiro de Código Fonte*.

✓ *Requisito de software 11*

Para efeito de processamento, o sistema deve ser capaz de seleccionar apenas a documentação interna referente ao cabeçalho do ficheiro de código fonte e aos cabeçalhos de cada elemento de código fonte.

Essa documentação deve ser sempre constituída por uma instância de *Cabeçalho de Ficheiro de Código Fonte*, seguida de zero ou mais instâncias de *Cabeçalho de Elemento de Código Fonte*².

Para efeito de processamento, a sintaxe destes cabeçalhos é a seguinte³:

```
⟨header⟩ ::= ⟨ret⟩  
  
{ ; }3 { - }77 ⟨ret⟩  
  
{ {⟨ASCII-char-except-ret⟩}* ⟨ret⟩}*  
  
{ ; }3 [⟨space⟩] Description {⟨space⟩}* ⟨ret⟩  
  
{ {⟨ASCII-char-except-ret⟩}* ⟨ret⟩}*  
  
{ ; }3 { - }77
```

⟨ASCII-char-except-ret⟩ ::= um carácter do conjunto ASCII, com excepção do carácter “quebra de linha” (código 10 do conjunto ASCII)

⟨space⟩ ::= ⟨sp⟩ | ⟨tab⟩

⟨sp⟩ ::= o carácter “espaço” (código 32 do conjunto ASCII)

² Deve ser observado que, no que diz respeito à identificação do texto a ser processado, o primeiro cabeçalho que estiver incluído num dado ficheiro de código fonte será sempre interpretado como um cabeçalho de ficheiro, independentemente dos caracteres existentes em {⟨carácter ASCII⟩}+.

⟨tab⟩ ::= o carácter “tabulação” (código 9 do conjunto ASCII)

⟨ret⟩ ::= o carácter “quebra de linha” (código 10 do conjunto ASCII)

Restrição

No lado direito da regra para o símbolo não terminal ⟨header⟩, se a primeira ocorrência de { ; } 3 { - } 77 tiver o seu primeiro carácter coincidente com primeira posição do ficheiro de código fonte, é aplicável um conjunto de regras diferente do anterior. O novo conjunto de regras é obtido a partir do primeiro suprimindo a primeira ocorrência de ⟨ret⟩ no lado direito da regra para o símbolo não terminal ⟨header⟩.

Nota sobre *design*

Embora existam outras alternativas à sintaxe apresentada (isto é, que traduzam o mesmo resultado), em particular para o símbolo não terminal ⟨header⟩, esta já tem em conta algumas decisões tomadas no âmbito do *design* do sistema, nomeadamente a leitura dos caracteres do ficheiro de código fonte ser feita no contexto da linha do ficheiro à qual pertencem.

Em termos de execução, o sistema começa por segmentar os caracteres do ficheiro em linhas. Após identificar as linhas (que constituem cadeias separadas pelo símbolo ⟨ret⟩), lê os caracteres de cada uma, individualmente.

1.3.3. Ficheiro de código fonte

A Tabela 1 descreve os possíveis tipos de atributos de um ficheiro de código fonte. A coluna “Fonte” indica qual tipo de fonte a partir do qual se pode obter o atributo em causa, se directamente da documentação interna, ou de forma indirecta. A coluna “Obrigatório?” indica se há obrigatoriedade para a existência de um valor para esse atributo. A coluna “Chave” indica qual a cadeia de caracteres que, no cabeçalho do ficheiro, deve anteceder o valor desse atributo.

³ Na realidade, um cabeçalho de ficheiro é normalmente iniciado com outro tipo de informação, como por exemplo, *copyright*, que não é tida em conta para efeito de processamento da documentação interna.

1.3.3.1. Tipos de atributos

Nome	Descrição	Fonte	Obrigatório?	Chave
Description	Descreve o propósito do ficheiro do ponto de vista do utilizador (<i>o que faz</i>)	Doc. interna	Sim	Description
Date	Identifica a data da última alteração feita ao ficheiro	Doc. interna	Sim	Não tem
Example	Um ou mais exemplos sobre como usar o ficheiro	Doc. interna	Não	\example
Implementation notes	Descreve aspectos relacionados apenas com a implementação (<i>como faz</i>)	Doc. interna	Não	\implem-notes
References	Referencia outros ficheiros que estão relacionados com o ficheiro documentado.	Doc. interna	Não	\refs
Remarks	Descreve efeitos secundários ou outra informação que seja relevante para o uso do ficheiro	Doc. interna	Não	\remarks

Tabela 1: Tipos de atributos de um Ficheiro de Código Fonte

1.3.3.2. Documentação interna

✓ *Requisito de software 12*

O sistema deve ser capaz de processar a documentação interna existente num cabeçalho de ficheiro de código fonte.

Para efeito de processamento, a estrutura dessa documentação é a seguinte:

`<file-header> ::= <ret>`

`{ ; } 3 { - } 77 <ret>`

`{ ; } 3 [<space>] Description { <space> } * <ret> <doc-text> <ret>`

`[{ ; } 3 { <space> } + \example { <space> } * <ret> <doc-text> <ret>]`

`[{ ; } 3 { <space> } + \remarks { <space> } * <ret> <doc-text> <ret>]`

`[{ ; } 3 { <space> } + \refs { <space> } * <ret> <doc-text> <ret>]`

`[{ ; } 3 { <space> } + \implem-notes { <space> } * <ret> <doc-text> <ret>]`

{ ; } 3 [<space>] History { <space> } * <ret>

{ ; } 3 { <space> } + Date { <space> } + Author { <space> } + Description
{ <space> } * <ret>

<doc-text> <ret>

{ ; } 3 { - } 77 <ret>

<space> ::= <sp> | <tab>

<sp> ::= o carácter “espaço” (código 32 do conjunto ASCII)

<tab> ::= o carácter “tabulação” (código 9 do conjunto ASCII)

<ret> ::= o carácter “quebra de linha” (código 10 do conjunto ASCII)

Os atributos são obtidos conforme se indica na Tabela 2.

Tipo de atributo	Fonte
Description	O valor do atributo é dado pelo valor de <doc-text> — suprimido de alguns caracteres — incluído na sequência Description { <space> } * <ret> <doc-text> <ret>
Example	O valor do atributo é dado pelo valor de <doc-text> — suprimido de alguns caracteres — incluído na sequência \example { <space> } * <ret> <doc-text>
Implementation notes	O valor do atributo é dado pelo valor de <doc-text> — suprimido de alguns caracteres — incluído na sequência \imlem-notes { <space> } * <ret> <doc-text>
References	O valor do atributo é dado pelo valor de <doc-text> — suprimido de alguns caracteres — incluído na sequência \refs { <space> } * <ret> <doc-text>
Remarks	O valor do atributo é dado pelo valor de <doc-text> — suprimido de alguns caracteres — incluído na sequência \remarks { <space> } * <ret> <doc-text>
Date	O valor do atributo é dado pela última data (com o formato aa/mm/dd) existente na instância de <doc-text> incluída na seguinte sequência: Date { <space> } + Author { <space> } + Description { <space> } * <ret> <doc-text> <ret>

Tabela 2: Valores para os atributos de um Ficheiro de Código Fonte

Notas

No lado direito da regra para o símbolo não terminal <file-header>, se a primeira ocorrência de { ; } 3 { - } 77 tiver o seu primeiro carácter coincidente com primeira posição do ficheiro de código fonte, é

aplicável um conjunto de regras diferente do anterior. O novo conjunto de regras é obtido a partir do primeiro suprimindo a primeira ocorrência de <ret> no lado direito da regra para o símbolo não terminal <file-header>.

Nota sobre *design*

Embora existam outras alternativas à sintaxe apresentada, esta já tem em conta algumas decisões tomadas no âmbito do *design* do sistema, nomeadamente a leitura dos caracteres do ficheiro de código fonte ser feita no contexto da linha do ficheiro à qual pertencem.

Em termos de execução, o sistema começa por segmentar os caracteres do ficheiro em linhas. Após identificar as linhas (que constituem cadeias separadas pelo símbolo <ret>), lê os caracteres de cada uma, individualmente.

1.3.4. Elemento de código fonte

1.3.4.1. Tipos de atributos

A Tabela 3 descreve os possíveis tipos de atributos de um elemento de código fonte. A coluna “Comum?” indica se o tipo de atributo é um comum ou não a todos os constructores. A coluna “Fonte” indica qual tipo de fonte a partir do qual se pode obter o atributo em causa, se directamente da documentação interna, ou de forma indirecta. A coluna “Obrigatório?” indica se há obrigatoriedade para a existência de um valor para esse atributo. A coluna “Chave” indica qual a cadeia de caracteres que, no cabeçalho do elemento, deve anteceder o valor desse atributo.

Nome	Descrição	Comum?	Fonte	Obrigatório?	Chave
Arguments	Descreve o tipo e o propósito de cada argumento	Não	Doc. interna	Não	\args
Class attributes	Descreve cada atributo de uma classe	Não	Doc. interna	Não	\class-attribs
Description	Descreve o propósito do elemento do ponto de vista do utilizador (<i>o que faz</i>)	Sim	Doc. interna	Sim	Description
Example	Um ou mais exemplos sobre	Sim	Doc.	Não	\example

Nome	Descrição	Comum?	Fonte	Obrigatório?	Chave
	como usar o elemento		interna		
Implementation notes	Descreve aspectos relacionados apenas com a implementação (<i>como</i> faz)	Sim	Doc. interna	Não	\implem-notes
References	Referencia outros elementos que estão relacionados com o elemento documentado. Pode incluir outros documentos	Sim	Doc. interna	Não	\refs
Remarks	Descreve efeitos secundários ou outra informação que seja relevante para o uso do elemento	Sim	Doc. interna	Não	\remarks
Return types	Identifica o tipo dos possíveis valores devolvidos	Não	Doc. interna	Não	\return-types
Visibility	Usado para determinar em que manuais o elemento deve ser mostrado. É muito importante para facilitar a produção automática de diferentes manuais	Sim	Doc. interna	Sim	\visibility
Date	Identifica a data da última alteração feita ao elemento	Sim	Doc. interna	Sim	Não tem
Name	Nome do elemento. Para métodos pode incluir a sua assinatura	Sim	Doc. interna	Sim	Não tem
Category	Identifica a categoria do elemento (e.g., função, método, macro).	Sim	Doc. derivada	Sim	N/A
Class operations	Descreve as operações da classe (constructores de objectos, acessores, etc.)	Não	Doc. derivada	Não	N/A
Interface	Sub-conjunto das operações de uma classe que são visíveis para os utilizadores da classe	Não	Doc. derivada	Não	N/A
Package	Identifica o pacote no qual o elemento está definido	Sim	Doc. derivada	Sim	N/A

Nome	Descrição	Comum?	Fonte	Obrigatório?	Chave
Signature	Caracteriza a especialização de um método	Não	Doc. derivada	Não	N/A
Source	Identifica o ficheiro no qual o elemento está definido	Sim	Doc. derivada	Sim	N/A
Super classes	Identifica as super classes de uma classe	Não	Doc. derivada	Não	N/A
Syntax	Descreve as regras de utilização do elemento	Não	Doc. derivada	Sim	N/A
System	Sistema lógico ao qual o elemento pertence	Sim	Doc. derivada	Não	N/A
Value	Valor por omissão de uma variável, parâmetro ou constante	Não	Doc. derivada	Não	N/A

Tabela 3: Tipos de atributos de um Elemento de Código Fonte

1.3.4.2. Documentação interna

✓ Requisito de software 14

O sistema deve ser capaz de processar a documentação interna existente num cabeçalho de elemento de código fonte.

Para efeito de processamento, a estrutura dessa documentação é a seguinte:

⟨element-header⟩ ::= ⟨ret⟩

{ ; }3 { - }77 ⟨ret⟩

⟨doc-text⟩

{ ; }3 [⟨space⟩] Description {⟨space⟩}* ⟨ret⟩ ⟨doc-text⟩ ⟨ret⟩

{ ; }3 {⟨space⟩}+ \visibility {⟨space⟩}* ⟨ret⟩ ⟨doc-text⟩ ⟨ret⟩

{ [{ ; }3 {⟨space⟩}+ \args {⟨space⟩}* ⟨ret⟩ ⟨doc-text⟩ ⟨ret⟩]

[{ ; }3 {⟨space⟩}+ \return-types {⟨space⟩}* ⟨ret⟩ ⟨doc-text⟩ ⟨ret⟩]

|

[{ ; }3 {⟨space⟩}+ \class-attrs {⟨space⟩}* ⟨ret⟩ ⟨doc-text⟩ ⟨ret⟩] }

```
[ { ; } 3 { <space> } + \example { <space> } * <ret> <doc-text> <ret> ]
[ { ; } 3 { <space> } + \remarks { <space> } * <ret> <doc-text> <ret> ]
[ { ; } 3 { <space> } + \refs { <space> } * <ret> <doc-text> <ret> ]
[ { ; } 3 { <space> } + \implem-notes { <space> } * <ret> <doc-text> <ret> ]

{ ; } 3 [ <space> ] History { <space> } * <ret>

{ ; } 3 { <space> } + Date { <space> } + Author { <space> } + Description
{ <space> } * <ret>

<doc-text> <ret>

{ ; } 3 { - } 77 <ret>
```

<space> ::= <sp> | <tab>

<sp> ::= o carácter “espaço” (código 32 do conjunto ASCII)

<tab> ::= o carácter “tabulação” (código 9 do conjunto ASCII)

<ret> ::= o carácter “quebra de linha” (código 10 do conjunto ASCII)

Os atributos são obtidos conforme se indica na Tabela 4.

Tipo de atributo	Fonte
Arguments	O valor do atributo é dado pelo valor de <doc-text> — suprimido de alguns caracteres — incluído na sequência \args { <space> } * <ret> <doc-text>.
Class attributes	O valor do atributo é dado pelo valor de <doc-text> — suprimido de alguns caracteres — incluído na sequência \class-attrs { <space> } * <ret> <doc-text>
Description	O valor do atributo é dado pelo valor de <doc-text> — suprimido de alguns caracteres — incluído na sequência Description { <space> } * <ret> <doc-text> <ret>
Example	O valor do atributo é dado pelo valor de <doc-text> — suprimido de alguns caracteres — incluído na sequência \example { <space> } * <ret> <doc-text>
Implementation notes	O valor do atributo é dado pelo valor de <doc-text> — suprimido de alguns caracteres — incluído na sequência \implem-notes { <space> } * <ret> <doc-text>
References	O valor do atributo é dado pelo valor de <doc-text> — suprimido de alguns caracteres —

Tipo de atributo	Fonte
	incluído na sequência <code>\refs {<space>* <ret> <doc-text></code>
Remarks	O valor do atributo é dado pelo valor de <code><doc-text></code> — suprimido de alguns caracteres — incluído na sequência <code>\remarks {<space>* <ret> <doc-text></code>
Return types	O valor do atributo é dado pelo valor de <code><doc-text></code> — suprimido de alguns caracteres — incluído na sequência <code>\return-types {<space>* <ret> <doc-text></code>
Visibility	O valor do atributo é dado pelo valor de <code><doc-text></code> — suprimido de alguns caracteres — incluído na sequência <code>\visibility {<space>* <ret> <doc-text></code>
Date	O valor do atributo é dado pela última data (com o formato aa/mm/dd) existente na instância de <code><doc-text></code> incluída na seguinte sequência: <code>Date {<space>+ Author {<space>+ Description {<space>* <ret> <doc-text> <ret></code>
Name	O valor do atributo é dado pelo valor de <code><doc-text></code> — suprimido de alguns caracteres — incluído na sequência <code><separator> <doc-text></code>

Tabela 4: Valores para os atributos de um Elemento de Código Fonte

Nota sobre *design*

Embora existam outras alternativas à sintaxe apresentada, esta já tem em conta algumas decisões tomadas no âmbito do *design* do sistema, nomeadamente a leitura dos caracteres do ficheiro de código fonte ser feita no contexto da linha do ficheiro à qual pertencem.

Em termos de execução, o sistema começa por segmentar os caracteres do ficheiro em linhas. Após identificar as linhas (que constituem cadeias separadas pelo símbolo `<ret>`), lê os caracteres de cada uma, individualmente.

Glossário

Atributo de Elemento de Código Fonte

Constitui uma propriedade de um elemento de código fonte. Representa parte da informação que caracteriza esse elemento.

Atributo de Ficheiro de Código Fonte

Constitui uma propriedade de um ficheiro de código fonte. Representa parte da informação que caracteriza esse elemento.

Cabeçalho de Elemento de Código Fonte

Bloco de documentação interna que descreve informação sobre um elemento de código fonte, em particular os seus atributos. Quanto à sua localização num ficheiro de código fonte, o bloco é contíguo e precede o elemento de código fonte ao qual se refere.

Cabeçalho de Ficheiro de Código Fonte

Bloco de documentação interna que descreve informação sobre um ficheiro de código fonte, em particular os seus atributos. Constitui o primeiro bloco de documentação existente num ficheiro de código fonte.

Elemento de Código Fonte

É o resultado de um constructor.

