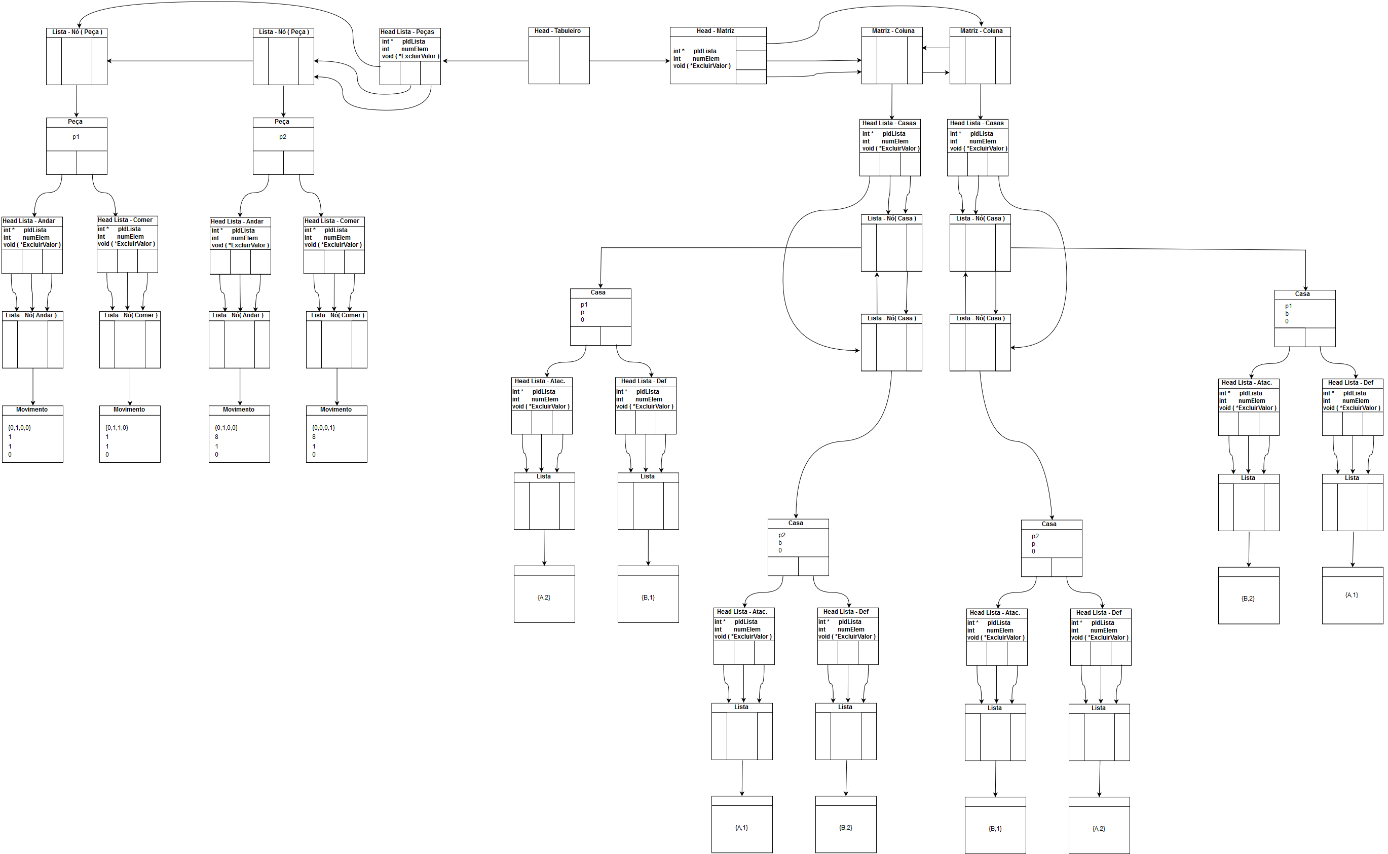
**Estrutura Unificada - Modelo e Assertivas**

**C:\Users\Lucas\Desktop\INF1301-T2.png**

**Estrutura Unificada – Exemplo**

****

**Tanto o modelo quanto o exemplo também serão encaminhados num PDF a parte para melhor visualização.**

**Arquitetura:**

**C:\Users\Lucas\Desktop\Arq (3).png**

**Argumentação de Corretude:**

**Função 1:**

**AE->**

  LIS\_tpCondRet LIS\_ObterElemento( LIS\_tppLista pLista, void \*\* pValor )

  {

     if ( pLista->pElemCorr == NULL )

     {

       \*pValor = NULL ;

            return LIS\_CondRetListaVazia;

     } /\* if \*/

**AI1->**

       \*pValor = pLista->pElemCorr->pValor;

**AI2->**

       return LIS\_CondRetOK;

  } /\* Fim função: LIS  &Obter referência para o valor contido no elemento \*/

**AS->**

**Argumentação de Sequência:**

**AE:**

* Lista não pode ser nulo, ou seja, existe e pode estar vazia.
* Valem as assertivas de Listas Duplamente Encadeadas.

**AI1:**

* Lista não pode ser nula, e nem vazia. Ou seja, ela existe e contém elemento(s).

**AI2:**

* pValor está preenchido com o valor do elemento corrente.

**AS:**

* Valem as assertivas de Listas Duplamente Encadeadas.
* pValor pode estar preenchido com o valor do elemento corrente, e neste caso o retorno é OK. Se a lista está vazia, o retorno é Lista Vazia.

**Argumentação de Seleção:**

**AE = AE**

**AS = AI1**

**1) AE && ( C == T ) + B => AI1**

* Pela AE, a lista pode ser vazia. Nesta condição, C == T então AS.
* Ao executar o bloco, é retornado Lista Vazio.

**2)  AE && ( C ==  F ) => AI1**

* Pela AE, a lista é uma lista válida e pode estar com o corrente apontado pro nó a ser alterado. Nesta condição, C == F então AS.
* Ao executar, AI1 deve ser válido

**Função 2:**

**AE->**

      int ValidarTipoPeca( LIS\_tppLista pLista, char \* nome )

      {

            tpPeca \* pPeca;

**AI1->**

            LIS\_tpCondRet CondRet;

**AI2->**

            CondRet = LIS\_AndarInicio( pLista );

**AI3->**

            while( CondRet == LIS\_CondRetOK )

            {

                   LIS\_ObterElemento( pLista, ( void \*\* ) &pPeca );

**AI5->**

                   if ( strcmp( nome, pPeca->nome ) == 0 )

                   {

                         return TRUE;

                   }

**AI6->**

                   CondRet = LIS\_IrProxElemento( pLista );

            }

**AI4->**

            return FALSE;

      }

**AS->**

**Argumentação de Sequência:**

**AE:**

* Lista não pode ser nulo, ou seja, existe e pode estar vazia.
* Para a lista, valem as assertivas de Listas Duplamente Encadeadas.
* Nome deve ser um ponteiro para char válido.

**AS:**

* Se o tipo de peça existir, retorna verdadeiro.
* Se o tipo de peça não existir, retorna falso.

**AI1:**

* A variável pPeca foi devidamente declarada e definida.

**AI2:**

* A variável CondRet foi devidamente declarada e definida.

**AI3:**

* O elemento corrente da lista é o primeiro.

**AI4:**

* O elemento corrente da lista é o último e o tipo de peça procurado não foi encontrado.
* O elemento corrente é o elemento procurado.

**Argumentação de Repetição:**

**AE =>** AI3

**AS =>** AI4

**AINV:**

* Existem dois conjuntos, a pesquisar e já pesquisado.
* O elemento corrente da lista aponta para o elemento a pesquisar.

**1) AE => AINV**

* Pela AE, o elemento corrente da lista é o primeiro. Neste caso, todos os elementos estão no conjunto “a pesquisar” e o conjunto “já pesquisado” está vazio. Logo a AINV é verdadeira.

**2) AE && ( C == F ) => AI4**

* Pela AE, o elemento corrente da lista é o primeiro. Logo, a lista pode estar vazia, valendo AI4.
* Pela AE, o elemento corrente da lista é o primeiro,  para o primeiro ciclo não concluir o tipo de peça deve ser encontrado na primeira ocorrência. Logo a função retorna verdadeiro e AI4 é válido.

**3) AE && ( C == T ) + B => AINV**

* Para a condição ser verdadeira, o primeiro elemento não pode ser o pesquisado. Neste caso, ele passa do conjunto “a pesquisar” para o “já pesquisado” e o elemento corrente da lista é reposicionado. Logo AINV é válida.

**4) AINV && ( C == T ) + B => AINV**

* Para garantir que AINV seja válida a cada ciclo, B garante que um elemento passe do conjunto a pesquisar para já pesquisado e o elemento corrente da lista seja reposicionado.

**5) AINV && ( C == F ) => AI4**

* O elemento foi encontrado. Logo AI4 é válida.
* Não existem mais elementos no conjunto “a pesquisar”. Logo AI4 é válida.

**6) Término**

* A cada ciclo um elemento é retirado do conjunto “a pesquisar” e colocado no conjunto “já pesquisado”. Como o número de elementos é finito, o número de passos necessários para terminar a repetição também é finito.

**Argumentação de Sequência:**

**AE = AS = AINV**

**AI5:**

* Como AINV é válido, o elemento corrente da lista é válido. Como o elemento corrente da lista é válido, seu valor deve estar em pPeca.

**AI6:**

* Retorna verdadeiro caso o elemento corrente seja o pesquisado.
* Elemento corrente é diferente do pesquisado.

**Argumentação de Seleção:**

**AE = AI5**

**AS = AI6**

**1) AE && ( C == T ) + B => AS:**

* Pela AE, pPeca possui valor igual ao do elemento a pesquisar. Como a condição é verdadeira, ambos são iguais e vale AS.

**2) AE && ( C == F ) => AS:**

* Pela AE, pPeca possui valor igual ao do elemento a pesquisar. Como a condição é falsa, o elemento a pesquisar e o pesquisado são diferentes, valendo AS.