4. Sistemática da Composição

Neste capítulo será apresentada uma sistemática de uso dos esquemas conceituais do Conteúdo, da Prática e da Composição, apresentados no Capítulo 3, bem como nas teorias de aprendizagem, abordagens pedagógicas e projeto instrucional.

4.1. Outras Informações Necessárias à Composição

A composição de SLOs com base em conteúdos e práticas também deve levar em consideração o "termo de entrada", as "meta-informações de entrada" e o perfil do professor, que serão descritos nas próximas seções.

4.1.1. Termo de Entrada

O "termo de entrada" é aquilo que se deseja buscar, representado através de um termo ou conjunto de termos necessários à localização do(s) objeto(s) componente(s). Exemplo: OLAP; ou Data Warehouse.

4.1.2. Meta-Informação de Entrada

As "meta-informações de entrada" são as características relativas ao que se pretende encontrar e ao seu uso, e tem como base o LOM (IEEE, 2002). Estas características serão comparadas com os metadados dos objetos componentes localizados.

São dados a serem informados pelo professor para que os mesmos auxiliem na seleção e recuperação dos OCs, SALOs e/ou SLOs, através de seu batimento com os metadados destes objetos.

- **Tipo de Interatividade**: Pode ser ativo, expositivo ou combinado (misto).

- **Aprendizado ativo**: aprendizado pelo fazer. Exemplo: exercício (quer encontrar solução); questionário (múltipla escolha ou descritivo);
- **Aprendizado expositivo**: aprendizado passivo. Ocorre quando a tarefa do aprendiz consiste principalmente da absorção do conteúdo exposto, geralmente através de texto, imagens ou som, como, por exemplo: documento hipertexto (ler, navegar) e vídeo (ver, rebobinar, iniciar, parar).
- **Aprendizado combinado**: resulta da combinação dos tipos de interatividade ativo e expositivo. Exemplo: documento hipermídia com *applet* de simulação.

- **Tipo de Recurso de Aprendizado**: Pode ser "exercício", "simulação", "questionário", "diagrama", "figura", "gráfico", "índice", "slide", dentre outros.
- **Densidade semântica**: É o grau de concisão de um objeto. A densidade semântica pode ser estimada em termos de seu tamanho, abrangência ou duração, com tempos bem definidos de exibição.
- **Contexto**: Indica o nível para o qual o objeto foi considerado. Pode ser "primeiro grau", "segundo grau", "superior", "treinamento" e "outros".
- **Faixa etária do público-alvo**: Este dado refere-se à idade mental, que pode ser diferente da idade cronológica, por exemplo, {"0-9"; "10-17"; e "18+"}.
- **Dificuldade**: É o grau de complexidade do objeto em relação ao público-alvo. Este público-alvo pode estar relacionado à faixa etária e ao grau de escolaridade definidos anteriormente. Esta dificuldade (complexidade) pode ser medida como: muito fácil; fácil; mediano; difícil; muito difícil.
- **Tempo de aprendizado**: Deve estar relacionado com faixa etária e ao grau de escolaridade do público-alvo, definidos anteriormente, além do grau de complexidade do conteúdo, como, por exemplo: "PT1H30M" e "PT1M45S", onde os dois primeiros dígitos indicam o idioma do objeto (PT = português); em seguida vêm a duração,

sendo H utilizado para hora, M para minutos e S para segundos. Assim, no exemplo acima tem-se um objeto no idioma português, com 1 hora e 30 minutos de duração e um outro também no idioma português com 1 minuto e 45 segundos de duração.

- **Idioma**: É a linguagem humana dominada pelo usuário alvo desse objeto, como, por exemplo: “en”, “fr”, “it”, “pt”.

4.1.3. Perfil do professor

O perfil do professor está associado à teoria de aprendizagem e abordagens pedagógicas que ele costuma adotar em suas aulas.

Ao considerar o perfil do professor, procura-se observar um conjunto de características que influenciam a forma como os professores planejam e desenvolvem seus conteúdos e ministram suas aulas. Considera-se que há uma tendência em adotar determinadas abordagens pedagógicas e seguir algumas teorias de aprendizagem. Deste modo, estes são os itens considerados no perfil do professor que norteiam o algoritmo da composição. Desenvolver melhor este perfil do professor faz parte dos trabalhos futuros.

4.2. Seleção dos Objetos

A busca e seleção de objetos devem levar em consideração o termo de entrada, as meta-informações de entrada e o perfil do professor. Há três etapas a serem executadas antes de se seqüenciar os objetos:

- Busca e Filtragem dos SLOs
- Busca e Filtragem dos SALOs
- Busca e Filtragem dos OCs

Os dois primeiros passos consistem em se buscar estruturas anteriormente definidas (ou seja, seqüências ou composições anteriormente especificadas). Com base no termo de entrada, há uma pesquisa nos SLOs e SALOs em uma tentativa de se encontrar o que é de interesse do professor e as meta-informações de entrada

são confrontadas com os metadados destes objetos, de modo a recuperar apenas o que é realmente relevante para o contexto de aprendizagem.

Caso sejam encontrados SLOs que atendam ao termo e meta-informações de entrada, então as estruturas definidas como SALOs participantes da composição destes SLOs são descartadas (ou seja, no segundo passo não são considerados os objetos que participam da composição de SLOs que já foram selecionados. Assim, não há uma repetição dos objetos selecionados). O mesmo ocorre com os OCs relativos aos SLOs e SALOs já selecionados.

4.2.1. Busca e Filtragem de OCcs

A busca de OCcs se dá através da comparação do "Termo de Entrada" com os OCcs armazenados. Com o resultado da busca processa-se a filtragem do conteúdo (ou OCcs) mais relevantes, ou de maior interesse, com base na comparação de seus metadados com as "meta-informações de entrada".

Além disso, é realizada uma comparação das "abordagens pedagógicas" de preferência do professor (através dos dados contidos em seu perfil) com os atributos tipo de interatividade e tipo de recurso de aprendizagem presentes nos metadados das OCcs resultantes da busca. Esta comparação segue o seguinte critério:

- Tipo de Interatividade.
- Tipo de Recurso de Aprendizagem.

4.2.2. Busca e Filtragem de OCps

A busca de OCps se dá através da comparação do "termo de entrada" com as práticas (OCps) armazenadas. Com base no resultado da busca, faz-se a comparação das abordagens pedagógicas preferidas pelo professor, encontradas em seu perfil, com os metadados dos OCps armazenados, com base nos atributos "Tipo de interatividade" e "Tipo de recurso de aprendizagem". Também é considerado o tipo específico de OCp, como por exemplo, 'simulação', 'experimentos', 'criação', 'produção', etc.

4.2.3 Sistemática para Batimento dos Conteúdos com as Práticas selecionadas

O batimento de OCcs e OCps de interesse se dá com base nas Teorias de Aprendizagem. Nesta tese trabalha-se com os três grupos considerados principais de Teorias de Aprendizagem, a saber: Comportamentalista, Cognitivista e Construtivista. Então, existem três abordagens a serem consideradas para o batimento do Conteúdo (OCcs) e Prática (OCps), após a busca e filtragem dos mesmos.

4.2.3.1. Sistemática para batimento dos Conteúdos e Práticas selecionadas segundo a Teoria de Aprendizagem Comportamentalista

Se a Teoria de Aprendizagem detectada no perfil do professor for a Comportamentalista, então selecionam-se as práticas e todos os seus respectivos conteúdos, ou seja, aqueles OCcs que estejam ligados às práticas selecionadas pelos relacionamentos '*precisa_d*e' ou '*é-compreendido_por*'.

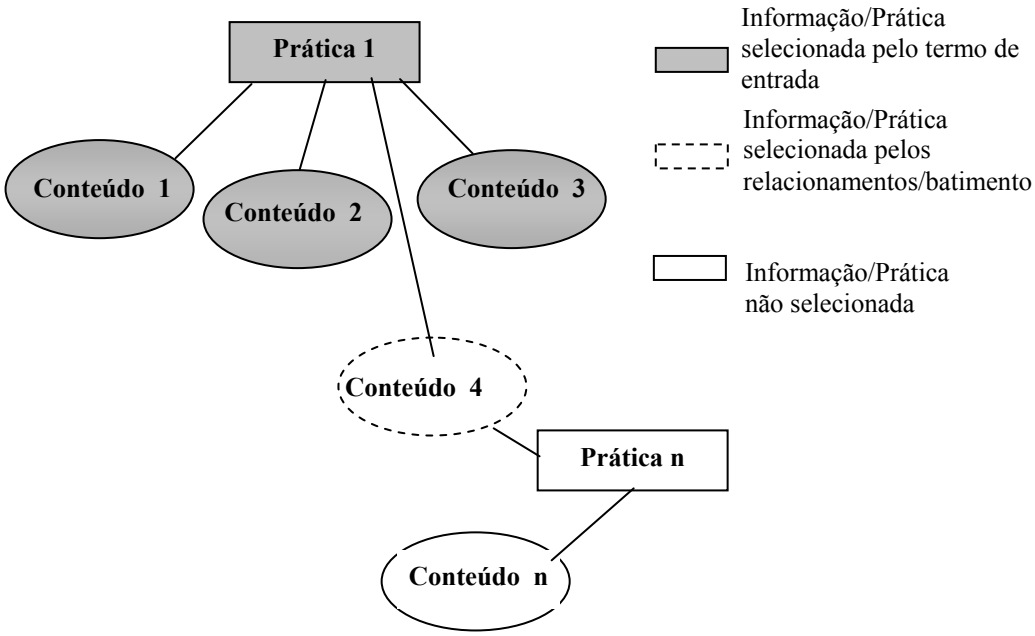

Figura 4.1: Batimento de OCps com OCcs (Teoria Comportamentalista)

A Figura 4.1 apresenta um exemplo com batimento segundo a teoria Comportamentalista. Neste exemplo, foram buscados e filtrados a Prática 1 e os Conteúdos 1 a 3, ou seja, eles foram selecionados anteriormente. Entretanto, como o Conteúdo 4 explica a Prática 1, então ele também é selecionado através do batimento. Os outros objetos que não tinham a ver com o termo de entrada e nem com batimento através dos relacionamentos não são selecionados.

4.2.3.2. Sistemática para batimento dos conteúdos e práticas selecionadas segundo a Teoria de Aprendizagem Cognitivista

Se a Teoria de Aprendizagem detectada no perfil do professor for Cognitivista, então selecionam-se o conteúdo e suas respectivas unidades conceituais, bem como as práticas que sedimentam este conteúdo, ou seja, aqueles OCps que estejam ligados aos conteúdos selecionados pelo relacionamento '*é-sedimentado_por*'. Além disso, verifica-se se todos os conteúdos relacionados às práticas recuperadas foram selecionados. Em caso positivo, então recuperam-se estas práticas e conteúdos. Caso contrário, esta prática não deve ser selecionada.

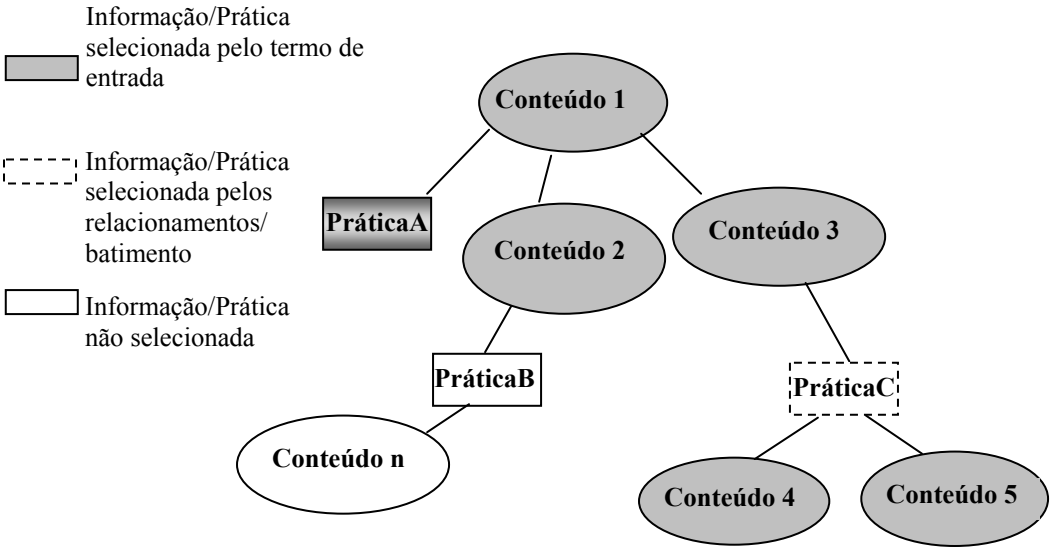

Figura 4.2 : Batimento de OCcs com OCps (Teoria Cognitivista)

A Figura 4.2 apresenta um exemplo um batimento segundo a teoria Cognitivista. Neste exemplo, foram buscados e filtrados os Conteúdos 1 a 5 e a Prática A, ou seja, eles foram selecionados anteriormente. Entretanto, como a Prática C sedimenta o entendimento dos Conteúdos 3, 4 e 5, então ela também é selecionada através do batimento. Vale ressaltar que a Prática B não foi selecionada no batimento porque o Conteúdo N não havia sido selecionado anteriormente.

4.2.3.3. Sistemática para Recuperar os Conteúdos e Práticas Selecionadas segundo a Teoria de Aprendizagem Construtivista

Se a teoria de aprendizagem detectada no perfil do professor for a Construtivista, então devem ser considerados os dois aspectos simultaneamente (Conteúdo e Prática). Todas as práticas relacionadas aos conteúdos selecionados, assim como todos os conteúdos relacionados às práticas selecionadas devem ser selecionados.

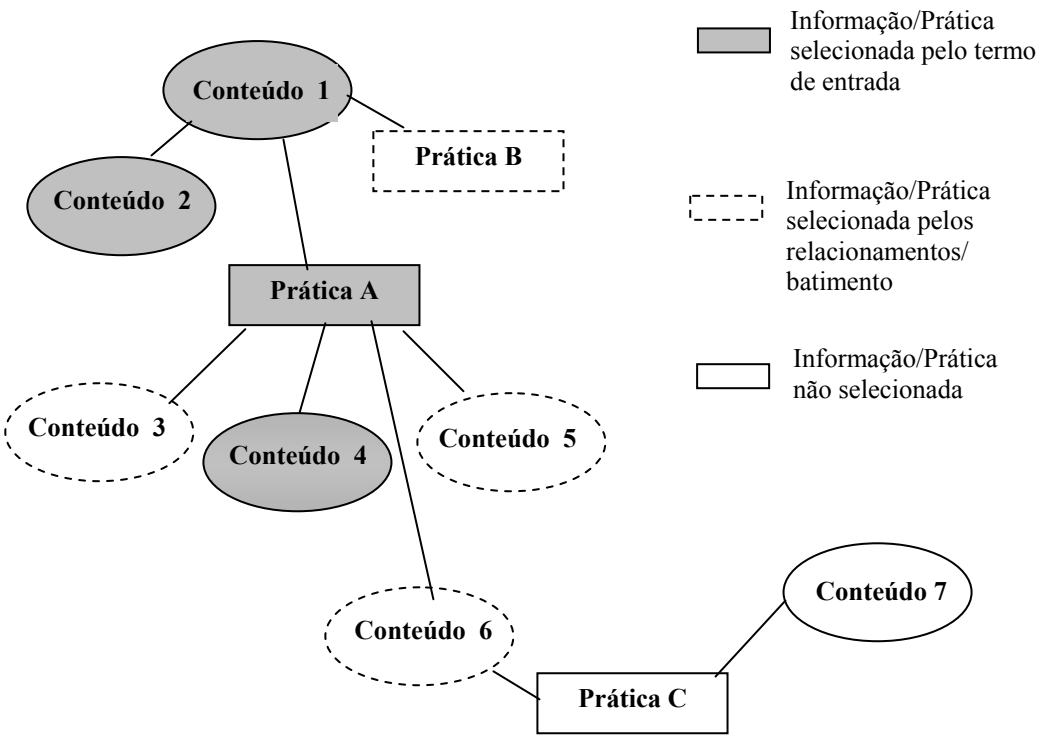

Figura 4.3 : Batimento de OCcs e OCps (Teoria Construtivista)

A Figura 4.3 apresenta um exemplo de batimento segundo a teoria Construtivista. Neste exemplo, foram buscados e filtrados os Conteúdos 1, 2 e 4, bem como a Prática A, ou seja, eles foram selecionados anteriormente. Entretanto, como a Prática B sedimenta o entendimento do Conteúdo 1, ela é selecionada no batimento. O mesmo acontece com os Conteúdos 3, 5 e 6, pois a Prática A precisa deles para ser entendida, ou é compreendida através desses conteúdos. Vale ressaltar que a Prática C não foi selecionada no batimento porque o Conteúdo 6 não havia sido selecionado anteriormente, o mesmo ocorrendo com o Conteúdo 7, já que a Prática C também não havia sido selecionada anteriormente.

4.3. Seleção de Unidades Conceituais e Ações

A teoria de aprendizagem também influencia a seleção de unidades conceituais. Assim, a seleção das unidades conceituais ocorre conforme a característica de cada teoria, logo:

4.3.1. Seleção de Unidades Conceituais

Se a teoria de aprendizagem é Comportamentalista
Então
As unidades conceituais a serem adotadas podem ser:
Introdução; Definição; Figura e Tabela de Procedimentos;
Senão, Se a Teoria de Aprendizagem é a Cognitivista
Então
As unidades conceituais a serem adotadas podem ser:
Introdução; Definição; Exemplo; Contra-Exemplo; Lista de Fatos; Tabela de Decisão; Demonstração e Orientação;
Senão, Se a Teoria de Aprendizagem é a Construtivista
Então
As unidades conceituais a serem adotadas podem ser:
Introdução; Definição; Exemplo; Tabela de Fatos; Tabela Combinada; Tabela de Estágios; Diagramas de Blocos; Declaração de Princípios; Orientação e Observação.

Similarmente, tem-se, para as ações, os procedimentos descritos na seção 4.3.2.

4.3.2. Seleção das Ações

Se a Teoria de Aprendizagem é Comportamentalista

Então

As ações a serem adotadas podem ser:

Ler; Ver; Ouvir; Produzir; Memorizar; Desenhar e Compor;

Senão, Se a Teoria de Aprendizagem é Cognitivista

Então

As ações a serem adotadas podem ser:

Ler; Ver; Ouvir; Ordenar; Selecionar; Analisar; Manipular; Modelar; Simular; Discutir; Apresentar; Debater; Criar; Memorizar; Desenhar e Compor;

Senão, Se a Teoria de Aprendizagem é a Construtivista

Então

As ações a serem adotadas podem ser:

Ler; Ver; Ouvir; Ordenar; Selecionar; Analisar; Manipular; Modelar; Simular; Discutir; Apresentar; Debater; Criar; Memorizar; Desenhar e Compor.

4.4. Sistemática de Seqüenciamento de OCs

Tendo selecionado os conteúdos e práticas, é necessário estabelecer o plano de seqüenciamento (plano de apresentação) dos objetos selecionados para o usuário final (professor). Dentro deste seqüenciamento é importante contemplar as possibilidades de organização entre Conteúdo e Prática que foram selecionados. Por convenção, serão considerados inicialmente todos os conteúdos necessários e em seguida as práticas, uma vez que esta é a estratégia geralmente adotada.

4.4.1. Algoritmo de Composição

Esta seção tem como objetivo especificar a composição de objetos componentes com base nos repositórios existentes. Assim, baseado em um algoritmo de seqüenciamento de objetos componentes, um professor pode compreender melhor a forma de implementar uma aula, um módulo ou um curso, reduzindo erros e eventuais omissões na implementação da solução.

Ressalta-se que as estruturas mais completas/complexas (SLOs e SALOs) definidas anteriormente não foram consideradas neste algoritmo porque são especificadas sem relacionamentos entre objetos do mesmo nível (isto é, não há relacionamentos entre SLOs e nem entre SALOs, exceto os de composição). Assim, poderiam ser tratados como objetos-raízes (que serão explicados posteriormente), sendo difícil definir sua posição correta em um seqüenciamento. Na aplicação da abordagem adotada na presente tese convenciona-se colocar estes elementos antes do restante do seqüenciamento. Assim, tem-se SLOs, SALOs e depois o restante do resultado do algoritmo de seqüenciamento.

Ao desenvolver o algoritmo de seqüenciamento de objetos componentes, alguns aspectos devem ser considerados, tais como:

- Quais objetos componentes foram selecionados?
- Quais destes objetos podem ser considerados como raiz para o seqüenciamento?

Assim, o primeiro passo no seqüenciamento é selecionar os objetos componentes de interesse de acordo com o perfil do usuário/professor, um termo de entrada e meta-informações de entrada, que indicam o que se deseja, de modo a formar o curso, módulo ou aula. Os objetos componentes resultantes da seleção são colocados em uma "sacola" (*bag*).

$$Bag = \{OCi\} \text{ onde } i: 1..n \text{ e } ((OC = OCc) \vee (OC = OCp)) \wedge (OCc \neq OCp).$$

Ou seja, uma sacola é um conjunto de n objetos componentes, que são do tipo conteúdo ou do tipo prática.

Em seguida, é necessário obter do "Bag", os objetos componentes do tipo raiz, que são os OCs que não seguem ninguém e/ou não dependem nem necessitam de ninguém para serem entendidos. Esta busca a OCs do tipo raiz é

realizada pela operação “SelectRootOCs()”, gerando uma lista denominada “RootOCsList”, conforme mostra a linha 02 do Quadro 4.1. Nesta lista podem ser encontrados tanto OCs Conteúdo (OCc), quanto OCs Prática (OCp).

Quadro 4.1: Algoritmo para Composição Objetos Componentes

Linha	Comando
01	**Início**
02	rootOCsList = selectRootOCs(); // Gera uma lista com os OCs raízes
03	OCsList = selectOCs(rootOCsList);
04	while (OCsList != null) **/* Processa as OCs */**
05	OCobj = getOC(OCsList);
06	ChildrenList = expand (OCobj);
07	while (ChildrenList != null)
08	Child = getChild(ChildrenList);
09	write(resultList,Child);
10	browsedOCsFromChild = browsing(Child,“é_detalhada_por”);
11	while (browsedOCsFromChild != null)
12	browsedOC = getOC(browsedOCsFromChild);
13	terminalObjList = expand (browsedOC);
14	while (terminalObjList != null)
15	terminalObj = getObj(terminalObjList);
16	write(resultList, terminalObj);
17	**Fim.**

Por convenção, a seqüência destes OCs raízes está condicionada à hierarquia de Tipo de Conteúdo (Conceito → Fato → Processo → Procedimento → Princípio) e de Tipo de Prática (Tratamento de Informação → Adaptação → Comunicação → Produção → Experimental). Considerando que qualquer OC deve ter associado uma Prática assimilativa, esta foi considerada como inerente aos próprios OCs. Assim, são processados os OCcs raízes seguidos dos OCps raízes. Conforme mostra a linha 03 do Quadro 4.1, a operação

"selectOCs(rootOCsList)" gera a lista "OCsList", a qual é ordenada segundo o seguinte procedimento:

Se Teoria de Aprendizagem for Cognitivista
    Então
        OCsTempList = {OCcs raízes → OCps raízes};
        Para cada OCi em OCsTempList faça
            Se OCi é OCc
                Então
                    OCsList = {OCsList → OCi → Relacionamentos (OCi, *é_relacionado_a*) → Relacionamentos (OCi, *é_sedimentado_por*)};
                Senão /* OCi é OCp */
                    OCsList = {OCsList → Relacionamentos (OCi, utiliza), Relacionamentos (OCi, *precisa_de*) → Relacionamentos (OCi, *é_compreendida_por*) → Relacionamentos (OCi, *é_complementada_por*) → OCi};
    Senão, Se Teoria de Aprendizagem for Comportamentalista
        Então
            OCsTempList = {OCps raízes → OCcs raízes};
            Para cada OCi em OCsTempList faça
                Se OCi é OCp
                    Então
                        OCsList = {OCsList → Relacionamentos (OCi, utiliza), Relacionamentos (OCi, *precisa_de*) → OCi → Relacionamentos (OCi, *é_complementada_por*) → Relacionamentos (OCi, *é_compreendida_por*)};
                    Senão /* OCi é OCc */
                        OCsList = {OCsList → Relacionamentos (OCi, *é_sedimentado_por*) → OCi → Relacionamentos (OCi, *é_relacionado_a*)}

Senão, Se Teoria de Aprendizagem for Construtivista

Então

/* Devido às características desta Teoria de Aprendizagem, não se especifica ordem, os objetos selecionados são apenas listados e deixa-se que o usuário escolha a seqüência */

Em seguida, é criado um laço onde a cada iteração é obtido um objeto da lista "OCsList" na linha 5 do Quadro 4.1, denominado "OCobj". A partir deste momento, caminha-se na árvore deste OC, isto é, expande-se este OC, obtendo assim suas unidades conceituais (OCucs), se ele for Occ, ou suas ações (OCacs), se ele for OCp.

expand (OC) = {OCuci} $\vee$ expand (OC) = {OCaci}, onde i: 1...n,

A operação "expand" retorna uma lista de OCucs relacionadas ao OCc ou uma lista de OCacs relacionadas ao OCp. Esta lista é denominada "ChildrenList", conforme mostra a linha 06 do Quadro 4.1, a qual é ordenada de acordo com o tipo do Conteúdo ou com o tipo de Prática. Por exemplo, o OCc do tipo conceito tem a seguinte ordem de suas OCucs: Introdução $\rightarrow$ Definição $\rightarrow$ Fato resumido $\rightarrow$ Exemplo $\rightarrow$ Contra-exemplo $\rightarrow$ Analogia $\rightarrow$ Observação, ao passo que o OCp do tipo Tratamento da informação tem a seguinte ordem de suas OCacs: Unir $\rightarrow$ Ordenar $\rightarrow$ Selecionar $\rightarrow$ Analisar $\rightarrow$ Manipular. Esta lista também é ordenada considerando-se os relacionamentos entre OCucs {*é_dependente_de*, *é_precedida_de*, *é_sucedida_por*} e aqueles entre OCacs {segue}.

Os objetos de "ChildrenList" (OCucs ou OCacs) são acessados seqüencialmente, sendo criado um segundo laço, conforme a linha 07 do Quadro 4.1. Cada objeto, obtido através da operação "getChild(ChildrenList)", deve ser escrito no seqüenciamento resultante através do comando write (resultlist,Child). À medida que os objetos componentes passam a fazer parte da lista do seqüenciamento resultante (resultList), este objeto recebe um flag indicando que ele já foi processado, evitando, assim, processamento redundante.

Em seguida, é verificada a existência de OCs relacionados ao objeto em Child, obtido da lista "ChildrenList", por meio da operação browsing (Child,"*é_detalhada_por*"), que verifica os relacionamentos do tipo *é_detalhada_por* para o objeto Child.

Os OCc obtidos na navegação, através desta operação de "browsing", formam uma lista "browsedOCsFromChild". É criado, então, um novo laço, onde cada OC da lista "browsedOCsFromChild" é expandido, obtendo as suas Unidades Conceituais (OCuc), conforme apresentado na linha 13 do Quadro 4.1, gerando a lista "TerminalObjList". O termo **Terminal** significa que não se desce mais um nível na árvore. Finalmente, cada OCuc da lista terminal é escrita no seqüenciamento resultante conforme descrito na linha 16 do Quadro 4.1.

Este seqüenciamento pode ser processado de formas diferenciadas, como segue:

- Seqüência ordenada de OCs;
- Seqüência não-ordenada de OCs;
- Seqüência em paralelo de OCs;
- Seqüências opcionais de OCs;
- Seqüência de OCs segundo uma decisão sobre uma condição;
- Seqüência com repetição de OCs enquanto uma condição for verdadeira.

4.4.2. Linguagem de Especificação do Seqüenciamento

Considerando a importância de se representar seqüências de OCs de modo não ambíguo e utilizar estruturas bem definidas para as formas de seqüenciamento, foi definida uma linguagem de especificação para o seqüenciamento de OCs/LOs, usando *Backus Naur Form* (BNF) (Garshol, 2003). Assim, a notação formal proposta é composta por um conjunto finito de regras e usa abstrações para representar estruturas sintáticas.

A linguagem proposta oferece ao projetista construtores para especificar:

- Expressões: necessárias para especificar condições;
- Fluxos (estruturas) de controle (seleção, repetição): necessárias para especificar ações sobre (sub) conjuntos de elementos (OCs);
- Regras (isto é, condições e ações): necessárias para definir estratégias de seqüenciamento;
- Estratégias de Seqüenciamento: necessárias para especificar exatamente qual e quando um OC deve ser executado (relativamente aos outros OCs).

Através desta linguagem são representadas seqüências ordenadas e não ordenadas, seqüências opcionais e em paralelo de OCs. Além disto, também são representadas condições, seleções e repetições de OCs.

Conforme dito anteriormente, a sintaxe da linguagem de especificação proposta é definida usando *Backus Naur Form* (BNF) (Garshol, 2003). As palavras reservadas (palavras-chave) da linguagem estão em negrito. Além disso, são utilizados alguns símbolos para agrupamentos, tais como '**{**', '**}**', '**[**', '**]**', e '/', '\'. Sempre que a expressão começar com o símbolo '**{**', obrigatoriamente deve existir o fechamento com o símbolo '**}**', o mesmo acontecendo com os pares de símbolos '**[**' e '**]**', '/' e '\'. Pode existir aninhamento destes símbolos (exemplo: **[**OC_1, **{**OC_2, OC_3**}**, OC_4**]**), mas não intercalação (exemplo: **[**OC_1, **{**OC_2,OC_3**]**, OC_4**}** não é uma expressão válida na linguagem).

Uma seqüência de OCs pode ser ordenada ou não ordenada (Pereira, 2004). No primeiro caso, representam-se os OCs delimitados pelos símbolos '**[**' e '**]**', para indicar que eles serão executados numa seqüência ordenada. Por exemplo, poder-se-ia ter a seguinte expressão definindo a seqüência dos OCs: **[**OCc1, OCc2, OCp1, OCc3, OCp2**]**, indicando que ter-se-iam dois conteúdos (OCc1, OCc2), sendo seguidos de uma prática (OCp1), e depois mais um conteúdo e uma prática (OCc3, OCp2).

Uma seqüência não ordenada é representada com os OCs sendo delimitados pelos símbolos '**{**' e '**}**'. Por exemplo, a expressão **{**OCp2, OCp1, OCp3**}** indica que a apresentação contendo práticas será executada em uma ordem arbitrária. Considerando as sintaxes para seqüências ordenadas e não ordenadas poder-se-ia ter uma seqüência contendo uma combinação de ambas as estratégias, por exemplo, **[{**$OCc1_1$, OCc2**}**, OCp1, OCc3, OCp2**]** indica que os OCs serão executados segundo uma determinada ordem, ou seja, tem-se duas definições/ dois conteúdos (OCc1e OCc2, cuja ordem de exibição é aleatória), seguidas de uma prática (OCp1), depois um outro conteúdo (OCc3) e finalmente uma nova prática (OCp2).

Outro exemplo seria: **{[**OCc2, OCp2**], [**OCc1, OCp1**], [**OCc3, OCp3**]}**, em que tem-se 3 subseqüências ordenadas de OCs **[**OCc2, OCp2**]**, **[**OCc1, OCp1**]** e **[**OCc3, OCp3**]** que poderão ser executadas em qualquer ordem, ou seja, estas três subseqüências serão executadas aleatoriamente, já que estão delimitadas por '**{**' e

‘**}**’. Entretanto, cada subseqüência tem seus OCs sendo executados em uma determinada ordem, ou seja, o primeiro conteúdo (OCc2) deve ser seguido por OCp2, ao passo que o segundo conteúdo (OCc1) deve ser obrigatoriamente executado antes de OCp1 e o terceiro conteúdo (OCc3) deve ser executado antes de OCp3. Neste exemplo, todos os conjuntos de OCs definidos entre chaves podem ser executados em qualquer ordem. Por outro lado, os OCs delimitados por colchetes devem ser executados na ordem em que foram especificados.

No caso de uma seqüência de OCs que serão executados ao mesmo tempo, ou seja, em paralelo, representam-se estes OCs delimitados pelos os símbolos **‘/’** e **‘\’** (Silva at al, 2005). Assim, a expressão /OCc2, OCc4\ indica, por exemplo, que um objeto componente conteúdo2 e um objeto componente conteúdo4 serão executados em paralelo. É interessante observar que esta execução em paralelo permite que diferentes OCs se complementem, originando execuções mais ricas e mais completas semanticamente. Uma aplicação óbvia desta execução em paralelo é a combinação de diversas mídias complementares, que consistem em diferentes versões de um mesmo conteúdo.

Além destas formas de seqüenciamento de OCs, pode-se observar ainda casos em que apenas uma quantidade determinada de OCs pertencentes a um grupo de OCs deve ser executada. Considerando a expressão 2 **{**OCp2, OCp3, OCp1, OCp4**}**, tem-se que 2 OCs contidos dentro do agrupamento de 4 OCs do tipo prática determinado pela expressão devem ser executados (neste exemplo, estes 2 OCs seriam executados em ordem aleatória). O agrupamento contém 4 diferentes práticas, mas apenas 2 destas práticas deverão ser executadas, segundo a escolha/interação do usuário.

Assim, define-se uma expressão de OC para especificar uma seqüência de OCs, utilizando os pares de símbolos ‘**[**’ e ‘**]**’, ‘**{**’ e ‘**}**’, e ‘/’ e ‘\’ com suas respectivas semânticas definidas anteriormente (Silva at al, 2005). Utilizando a BNF temos:

<Expressão_OC> **::=** < Expressão_OC>**(‘, ’**<Expressão_OC>**)***
| <OCs_Ordenados>**(‘, ’**<Expressão_OC>**)***
| <OCs _Não_Ordenados>**(‘, ’**<Expressão_ OC>**)***
| <OCs _Paralelos>**(‘, ’**<Expressão_ OC>**)***
| <OCs _Opcionais>**(‘, ’**<Expressão_ OC>**)***
| <OCuc>**(‘, ’**<Expressão_ OC>**)***

| <OCac>**(‘, ’**<Expressão_ OC>**)***

| <Seleção_OC>**(‘, ’**<Expressão_ OC>**)***

| <Repetição_ OC>**(**‘, ’<Expressão_ OC>**)***

Lembrando que o símbolo * em BNF representa zero ou mais ocorrências, tem-se que uma expressão de OC terá sempre uma forma que pode ser seguida ou não de outras expressões de OCs. Na primeira linha da definição de expressão de OC tem-se uma recursão, ou seja, uma expressão de OC podendo ser seguida de outras expressões de OC (desde que separadas por vírgulas).

Entretanto, pela definição acima, de Expressão_OC, torna-se necessário definir também a sintaxe de OCs_Ordenados, OCs_Não_Ordenados, OCs_Paralelos, OC_Opcional, Seleção_OC, Repetição_OC, OCuc e OCac.

Conforme explicado anteriormente, uma seqüência ordenada de OCs é expressa por OCs delimitados por colchetes. Assim, temos:

<OCs_Ordenados> **:=** ‘**[**’<Expressão_OC>‘**]**’

Similarmente, conforme explicado anteriormente, uma seqüência não ordenada de OCs é expressa por OCs delimitados por chaves, o que em nossa representação, seguindo a BNF, resulta em:

<OCs_Não_Ordenados> **::=** ‘**{**’<Expressão_ OC>‘}’

Foi explicado que a execução em paralelo de OCs seria representada com os OCs sendo delimitados por barras simples, resultando em:

<OCs_Paralelos> ::= ‘/’<Expressão_OC>‘\’

A representação de uma execução de uma quantidade *x* de elementos de um grupo de *y* elementos, sendo $x \leq y$ é feita através de OC_Opcional. Entretanto, é interessante observar que esta especificação só faz sentido para grupos de OCs e, portanto, estes possíveis grupos seriam dados por OCs_Ordenados, OCs_Não_Ordenados e/ou OCs_Paralelos. Assim, temos:

<OC_Opcional> ::= <numero>(<OCs_Ordenados>)

| <numero>(<OCs_Não_Ordenados>)

| <numero>(<OCs_Paralelos>)

<numero>::= <digito><digito>*

<digito>::= ‘0’ | ‘1’ | ‘2’ | ‘3’ | ‘4’ | ‘5’ | ‘6’ | ‘7’ | ‘8’ | ‘9’

<OCuc> é uma identificação do OCuc sendo referenciado. Podem ser utilizadas formas de identificação como, por exemplo, oids se for implementado como OWL.

<OCac> é uma identificação do OCac sendo referenciado. Podem ser utilizadas formas de identificação como, por exemplo, oids se for implementado como OWL.

Finalmente, o restante da especificação de uma expressão de seqüências de OCs representa estruturas de controle, ou seja regras (condições e ações) e repetições. Antes de apresentar a especificação de tais estruturas, descreve-se a especificação de condições.

As condições são especificadas de maneira similar ao que acontece em linguagens de programação. Operadores Booleanos são avaliados na seguinte ordem: ‘**NOT**’, ‘**AND**’ e ‘**OR**’. Observe também que operações aritméticas entre operandos estão previstas (‘+’, ‘**-**’, ‘*****’, ‘÷’, ‘**mod**’ e ‘div’). No contexto do presente trabalho, uma <Condição> é definida como uma expressão lógica.

<Condição> **::=** <Operando> (<Operação> <Operando>)*

<Operando> **::=** <Condição>

|‘**NOT**’ <Operando>

|<Função>

<Operação> **::=** ‘**>**’ | ‘**< **’ | ‘**=**’ | ‘**>=**’ | ‘**<=**’ | ‘**!=**’ | ‘**OR**’ | ‘**AND**’ | ‘**+**’ | ‘*****’ | ‘**-**’ | ‘**÷**’ | ‘**mod**’ | ‘**div**’

<Função> **::=** é uma chamada a um procedimento que é executado com base nas propriedades de OCs e retorna um valor

É importante ressaltar que para comparar dois operandos eles devem ser do mesmo tipo. Além disto, todos os procedimentos que possam ser especificados através de <Função> deverão ser listados antes da especificação de seqüências de OCs, de modo a tornar possível a validação das especificações.

Para especificar que uma ação (a execução de uma expressão de OCs) é dependente de uma condição, utiliza-se “Seleção_OC”, que é apresentada abaixo. A “Seleção_OC” é especificada de maneira similar ao que acontece em

linguagens de programação. Observe que, segundo esta especificação, pode-se ter uma cláusula indicando o que fazer caso a condição não seja válida.

<Seleção_OC> ::= '**If**' <Condição> '**Then**' <Expressão_OC>
| '**If**' <Condição> '**Then**' <Expressão_OC> ','
'**Else**' <Expressão OC>

Exemplos:

<Seleção_OC>::= **If (**TempExecução(OCc2)+TempExecução(OCp2))<50
Then [OCc2, OCp2**]**

<Seleção_OC>::= **If** (TempExecução(OCc2)+TempExecução(OCp2))<50
Then [OCc2, OCp2**], Else [**OCc12, OCp12**]**

Para este exemplo temos:

1. OCs: OCc2 (contendo uma definição de conjuntos); OCc12 (contendo uma segunda definição de conjuntos); OCp2 (contendo exercícios de conjuntos); OCp12 (contendo um teste sobre conjuntos e rastreando as interações do usuário).
2. Função: TempExecução recebe como parâmetro a identificação de um OC (OCc ou OCp) e retorna o tempo de execução destes OCs em minutos, seja ele Conteúdo ou Prática.

Assim, se o tempo de execução (OCc2) + (OCp2) for menor que 50, então deverão ser executados os OCs: OCc2 e OCp2, nesta ordem. Caso contrário, serão executados os OCs: OCc12 e OCp12.