ÍNDICE

1. INTRODUÇÃO

Neste relatório técnico tratamos de uma técnica de representação do conhecimento denominada *padrões*. Sua idéia original é atribuída ao arquiteto Christopher Alexander, que no final da década de 60 e na década de 70, propôs uma linguagem de padrões para utilização no projeto arquitetônico, com o objetivo de sanar diversos problemas por ele apontados nestes projetos (LEA, 1994). A *linguagem de padrões*[1] proposta (vide seção 2.3) inicia num nível de abstração muito grande, explicando como o mundo deveria ser dividido em nações, nações em regiões, e assim por diante, descendo até o nível de projeto de estradas, estacionamentos, comércio, lugares de trabalho, casas e templos de adoração. Os padrões vão apresentando foco com nível de detalhe cada vez maior, passando por uma discussão de como arranjar os quartos e salas de casas, até finalmente descrever o tipo de material a ser usado em paredes, a decoração da casa e como tratar a iluminação.

A forma e as características de padrões, bem como os métodos e processos associados, não são de modo algum especiais apenas para projetos arquitetônicos. A interconexão cuidadosa entre contextos, forças do espaço de problemas e soluções construtivas formam uma base ideal para a captura de outros tipos de conhecimento de projeto e prática. GAMMA *et al.* (1995) afirmam que, apesar de se referir a prédios e cidades, o que Alexander diz é verdade também em relação aos padrões de projetos orientados a objetos, se diferenciando apenas por apresentar as soluções em termos de objetos e interfaces, em vez de paredes e portas.

Para Alexander, um padrão é ao mesmo tempo uma coisa e a regra que diz como criar esta coisa e quando devemos criá-la. Ralph Johnson afirma que os padrões de Alexander aparentam ser um bom modelo para os tipos de planos necessários para o desenvolvimento de software (JOHNSON, 1994a). Na Ciência da Computação, padrões têm sido alvo de estudo desde o final da década de 80, tendo como inspiração o trabalho de Alexander (JOHNSON, 1994b).

[1] Um conjunto de padrões, cada um descrevendo como resolver um tipo específico de problema (vide seção 1.4).

No capítulo introdutório da publicação dos trabalhos apresentados na primeira conferência sobre padrões, a PLoP'94[2] (COPLIEN, 1994a), Ralph Johnson apresenta como uma justificativa da inauguração desta conferência a contínua insatisfação do grupo fundador com a literatura sobre desenvolvimento de software. Segundo Johnson, a tendência a sempre se privilegiar o novo fomenta uma rejeição pelo usual, não importando o quão útil possa ser. O grupo, pela sua experiência em Orientação a Objetos e Reutilização, sabe que projetos falham, mesmo utilizando o que há de mais moderno em tecnologia. Portanto, decidiram que era necessário concentrar a atenção na disseminação de soluções.

A motivação do grupo fundador dos seminários anuais PLoP ilustra bem o objetivo do uso de padrões. Como afirma JOHNSON (1994a), especialistas em projetos têm uma grande quantidade de padrões na cabeça e utilizam-nos para produzir projetos bem acabados, combinando-os e estendendo-os. O emprego de padrões tem sido, portanto, no sentido de disseminar soluções, incorporando, além de soluções, informações tais como descrições do problema e contexto de aplicação. Dessa maneira, a experiência de especialistas passa a estar disponível para o uso disseminado, fornecendo solução para problemas que assolam o desenvolvimento de software.

O objetivo principal dos fundadores da PLoP foi a criação de um manual para engenheiros de software que informe como projetar software, para ser usado de maneira semelhante aos de outras áreas de engenharia. A meta era permitir que um engenheiro de software realizasse seu trabalho de forma semelhante a um engenheiro civil, por exemplo, ao projetar uma ponte. Através da consulta a manuais específicos, ele seleciona um tipo de ponte (por exemplo, estrutura metálica ou concreto), determina a estrutura, os materiais, a forma, etc.

Na figura 1, encontra-se um gráfico estabelecendo uma escala de evolução de conceitos sobre a representação de dados com que se lida no desenvolvimento de software. O dado seria uma porção do mundo representado. A informação seria porções relacionadas. O conhecimento seria a obtenção de nova porção a partir da Informação existente. Finalmente, sabedoria seria a possibilidade de refletir, num novo contexto, um conhecimento existente (PRESSMANN, 1992). O uso dos padrões facilita a efetivação

[2] Realizada anualmente, desde então, dentro da conferência Conference on Object-Oriented Programming, Systems, Languages, and Applications - OOPSLA.

da sabedoria. Até agora, o trabalho de especialistas só estava disponível através de consultas a manuais que explicavam métodos de desenvolvimento, pela investigação de produtos acabados ou consultas aos próprios especialistas. O uso de padrões pretende tornar mais acessível este conhecimento.

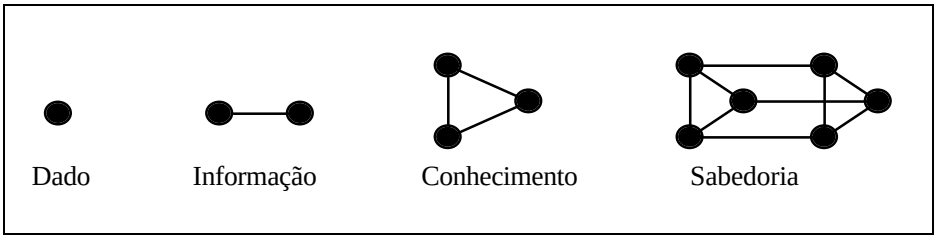

Figura 1 - Evolução de Conceitos do Desenvolvimento de Software (PRESSMANN, 1992)

LEA (1994), na análise que faz das possibilidades da conjugação dos conceitos de orientação a objetos e padrões, lembra que a maior parte dos trabalhos de pesquisa ainda estão na fase exploratória, e que, embora modelos de processos orientados a objetos ainda não estejam maduros, sua sinergia com modelos baseados em padrões é óbvia. Acaba por sugerir que é possível que as idéias, que isolaram Alexander do ramo principal da arquitetura comercial, podem acabar por encontrar seu maior e mais duradouro impacto na engenharia de software orientado a objetos.

ALFRED e MELLOR (1995) chegaram a afirmar que padrões iriam ser a "revolução industrial" do desenvolvimento de software. Afirmaram que seria possível, nos anos que faltam para o ano 2000, adotar um processo de desenvolvimento baseado em padrões com as fases de projeto e implementação automatizadas, mesmo considerando ser necessário ainda bastante trabalho para identificar, catalogar e representar padrões de análise, de projeto e de arquitetura de sistemas, bem como para definir formalismos para classificar restrições de aplicações e especificar políticas de desenvolvimento. No entanto, isto parece ser demasiadamente otimista.

PREE (1994) e GAMMA *et al.* (1995) consideram os padrões como uma extensão dos métodos de desenvolvimento de software orientados a objetos. COPLIEN e SCHMIDT (1995) consideram os padrões como uma técnica importante para o aperfeiçoamento da qualidade do software.

A pesquisa sobre padrões amadurece rapidamente. Sua introdução encontrou uma comunidade ansiosa por discutir soluções, na medida em que os problemas já eram

de conhecimento generalizado, bem como as ferramentas para lidar com eles. Dessa forma, na segunda conferência sobre padrões, a PLoP'95, observou-se que o valor de um padrão não mais estava na sua descoberta, mas em sua relevância, qualidade e impacto (VLISSIDES *et al.*, 1996).

Nos trabalhos aqui abordados, encontramos padrões aplicáveis ao longo de todo o processo de desenvolvimento de software. Há padrões específicos para a análise de requisitos, para o planejamento e gerência do processo, bem como, com bastante abundância, para o projeto de software.

A seguir, na seção 2, apresentamos uma conceituação teórica sobre os padrões. Na seção 3, apresentamos exemplos encontrados na literatura. Na seção 4, discutimos o desenvolvimento de software utilizando padrões. Por fim, na seção 5, concluímos o relatório técnico.

2. O QUE SÃO PADRÕES

Nesta seção é apresentada a conceituação de padrões, algumas possíveis aplicações, uma discussão sobre como encontrá-los e organizá-los, bem como as diferentes formas de como são definidos na literatura. Discute-se, ainda, sobre as diferenças, pouco claras em alguns trabalhos, entre padrões e frameworks. Por fim, tratamos da questão da classificação dos padrões.

2.1. Conceituação

Segundo GAMMA *et al.* (1995), Alexander observa que cada padrão descreve um problema que ocorre repetidamente em nosso meio, e inclui uma solução de forma genérica para o mesmo, de tal maneira que se pode usar esta solução mais de um milhão de vezes, sem nunca fazê-lo de maneira idêntica. LEA observa que alguns padrões são simplesmente receitas, mas que, apesar disso, um especialista pode usá-lo da mesma maneira que um cozinheiro usa uma receita, como auxílio na criação de uma visão pessoal de uma realização particular, mas ainda mantendo os ingredientes e proporções que são críticos (LEA, 1994).

A forma dos padrões é importante (uma discussão mais aprofundada pode ser encontrada na seção 2.5). Uma forma simples para padrões foi apresentada por LEA (1995), afirmando ser esta a mais comum dos padrões de Alexander:

SE
    você se encontra no contexto CONTEXTO,
    por exemplo, EXEMPLOS,
    com PROBLEMAS,
    conseqüentes às FORÇAS
ENTÃO
    por algumas RAZÕES,
    aplicar FORMATO E/OU REGRA DE PROJETO
    para construir SOLUÇÃO
    levando a NOVO CONTEXTO e OUTROS PADRÕES

Os padrões de GAMMA *et al.* (1995) apresentam uma forma genérica, diferindo um pouco da de Alexander. Os autores consideram os quatro elementos seguintes como essenciais para padrões de projeto:

- **Nome:** Uma identificação, de uma ou duas palavras, que se possa usar para descrever o problema, suas soluções e conseqüências. A nomeação de padrões aumenta o vocabulário de projeto e permite que se projete num nível de abstração mais alto. Além disso, a existência de um vocabulário de padrões facilita sua comunicação e, portanto, sua discussão e evolução;
- **Problema:** Descreve quando o padrão é aplicável, além de explicar o problema e o contexto. Pode descrever problemas específicos de projeto, como por exemplo, de que forma representar algoritmos como objetos. Pode, também, descrever estruturas de objetos e classes que são sintomáticos de um projeto sem flexibilidade. Pode, às vezes, incluir uma lista de condições para a aplicação do padrão;
- **Solução:** Descreve os elementos que compõem o projeto, seus relacionamentos, responsabilidades e colaborações. Não descreve um projeto concreto ou uma implementação em particular, já que um padrão é como um "template", que pode ser aplicado em várias situações diferentes. Ao contrário, um padrão apresenta uma descrição abstrata de um problema de projeto e como um arranjo genérico de elementos (classes e objetos) o soluciona; e
- **Conseqüências:** São os resultados e comprometimentos ("trade-offs") da aplicação do padrão. Apesar das conseqüências não serem consideradas quando se descreve decisões de projeto, elas são críticas para a avaliação de alternativas, bem como para compreensão de custos e benefícios da aplicação do padrão. As conseqüências em software quase sempre se referem a comprometimentos ('trade-offs") de espaço e tempo, mas podem abordar também questões de linguagem e implementação. Como a reutilização é sempre um fator importante no projeto orientado a objetos, as conseqüências de um padrão devem incluir o impacto na flexibilidade, estensibilidade e portabilidade do sistema.

LEA (1994), analisando o trabalho de Alexander, ressalta que os padrões devem conter uma descrição clara o suficiente para permitir que todos os participantes do desenvolvimento a entendam, e não somente profissionais experientes. Neste trabalho, LEA descreve as propriedades que os padrões de Alexander possuem, sendo as principais:

- **Encapsulamento:** Cada padrão encapsula um problema/solução bem definido. Padrões são independentes, específicos e formulados precisamente (o suficiente para deixar claro quando são aplicáveis e se capturam problemas relevantes). Além disso, deve ser assegurado que o padrão faça sentido por si só e que seja útil na construção de uma entidade reconhecível;
- **Abstração:** Padrões representam abstrações de experiência empírica e conhecimento do dia a dia. São genéricos dentro do contexto especificado, mas não necessariamente universais;
- **Abertura:** Padrões podem ser arbitrariamente estendidos em níveis de maior detalhe. Por exemplo, os padrões escolhidos para atacar uma parte de um projeto podem requerer sub-padrões; e
- **Composição:** Padrões são relacionados hierarquicamente. Padrões de maior granularidade são estabelecidos sobre, se relacionam com, e restringem os de menor granularidade.

Os padrões de COAD (COAD, 1992, COAD *et al.*, 1995) são definidos como uma combinação de um pequeno grupo de classes e objetos com relacionamentos entre eles, ocorrendo repetidamente em diferentes modelos de análise e projeto orientados a objetos, e aplicáveis múltiplas vezes numa mesma aplicação. COAD observa que métodos orientados a objetos já enfatizam certos padrões de relacionamento, como generalização-especialização, todo-parte, associação e troca de mensagens, que amarram os blocos de construção de mais baixo nível. Observa ainda que também já foram investigados *frameworks de aplicação*[3], mas que combinações de classes, objetos e relações entre classes e objetos, como encontradas nos padrões, ainda não haviam sido investigadas (a diferença entre frameworks e padrões é tratada na seção 2.4).

2.2. Como Encontrar Padrões

A descoberta de padrões exige um esforço duplo. É necessário ter a capacidade de identificar os problemas recorrentes, além de elaborar uma descrição efetiva do problema, incluindo a solução e outras informações.

[3] Um esqueleto de classes, objetos e relacionamentos, agrupados para a construção de uma aplicação específica.

Segundo LEA (1994), a construção de padrões é como a realização de uma *análise de domínio*[4], um processo social interativo de coleta, compartilhamento e amplificação de experiência e conhecimento distribuídos. Algumas vezes padrões são construídos de forma mecânica, a partir de outros, juntando-os e/ou transformando-os para que se apliquem a um novo domínio. No entanto, alguns padrões são tão referentes a conceitos universais que surgem da introspecção e intuição. LEA afirma que heurísticas são também importantes.

Referindo-se a padrões para análise e projetos orientados a objetos, COAD (1992) afirma que padrões são encontrados por tentativa e erro e por observação. COAD *et al.* (1995) definem estratégias que poderiam ser consideradas como padrões de desenvolvimento de modelos de objetos. Dentre as estratégias propostas, COAD *et al.* apresentam algumas que visam a descoberta de novas estratégias e padrões. Para a descoberta de novas estratégias, sugere "1) Preste atenção a cada pequeno passo dado na construção de alguma parte do modelo de objetos. Que conselho poderia dar a outros para que consigam realizar a mesma tarefa com menos passos, em menos tempo, com mais regularidade?; e 2) Considere cada correção feita no modelo. Que conselho poderia dar para ajudar outros a evitar o erro?" Com relação à descoberta de novos padrões, sugere: "1) Verifique cada grupo de dois, três, quatro, n objetos que possuem interação; 2) Generalize os nomes de cada participante; e 3) Relacione-os, por analogia, a outros domínios. Verifique se pode ser usado repetidamente."

2.3. Linguagem de Padrões - A Organização dos Padrões

O conjunto de padrões criado é usualmente denominado de *linguagem de padrões* (termo introduzido por Alexander). A linguagem de padrões proposta por Alexander deveria ser utilizada numa seqüência pré-determinada, do primeiro ao último padrão, sendo um padrão estruturado de forma a conduzir ao seguinte (JOHNSON, 1992). Nas abordagens descritas neste trabalho, os padrões são apresentados numa organização menos estruturada, havendo, em alguns casos, apenas sugestões de uma possível utilização conjunta.

[4] Análise de Domínio é o processo de identificar e organizar o conhecimento sobre alguma classe de problemas - o domínio do problema - para dar suporte à descrição e solução destes (ARANGO e PRIETO-DÍAZ, 1991).

Sobre os padrões de Alexander, LEA (1994) observa que a maioria dos padrões podem ser compostos, assim como podem ser componentes, minimizando a interação com outros. Observa-se ainda que padrões levam a outros padrões, num mesmo nível ou num nível de detalhe maior (menor granularidade). A experimentação com possíveis variantes e o exame de relacionamentos entre padrões, que juntos formam o todo, adiciona restrições e ajustes, bem como especializações e refinamentos específicos à situação. Como os detalhes das instanciações de padrões estão encapsulados, podem haver variações dentro da faixa estabelecida pelas restrições.

BUSCHMANN e MEUNIER (1995) observam que um *sistema de padrões* (como denominam sua linguagem de padrões), para ser útil, precisa considerar todas as possíveis restrições relativas à combinação e composição dos padrões que compreende. Apesar de cada padrão endereçar uma questão de projeto em particular, isolada e completa, nenhum padrão pode ser visto como totalmente independente dos outros. Os autores observam que, na realidade, um padrão é parte de uma estrutura maior e que pode conter, estar contido, ou estar interligado com outros padrões, com os quais também interagirá. Portanto, os padrões, quando combinados corretamente, devem complementar-se, de modo a que não sejam introduzidos problemas, mais do que são resolvidos. Ressaltam, ainda, que devem ser elaboradas orientações sob dois aspectos:

- Cada padrão deve especificar com quais padrões pode ser combinado e como, fornecendo ainda instruções sobre sua integração em estruturas maiores, ou sua composição a partir de estruturas menores; e
- Estruturas apresentam complexidade por si só, independentemente das questões de projeto endereçadas pelos padrões que a compõem. Portanto, regras gerais e instruções devem ser especificadas para sua composição e manipulação, independentemente de orientações para seus padrões constituintes.

ZIMMER (1995) observa que os relacionamentos entre os padrões de projeto apresentados por GAMMA *et al.* (1995), são de três tipos: 1) um padrão usa um outro na sua solução; 2) uma das variações de um padrão usa um outro na sua solução; e 3) os padrões endereçam problemas semelhantes.

2.4. Frameworks x Padrões

JOHNSON (1992) apresenta uma discussão sobre a diferença entre padrões e frameworks. JOHNSON afirma que:

- **Framework:** É um projeto reutilizável em soluções de problemas em algum domínio específico.
- **Padrão:** São orientados a problemas e não a soluções. Cada padrão descreve como resolver uma pequena parte de um problema de projeto. Padrões são, perfeitamente, indicados para ensinar o uso de um framework.

Ao desenvolver um software, um framework é escolhido e, então, faz-se adaptações necessárias para que se torne útil. Padrões seriam úteis para este trabalho de adaptação, na medida em que indicassem a melhor forma de estruturação que se deseja (utilizando, por exemplo, os meta-padrões de PREE (1994)), bem como na melhora do projeto obtido após as adaptações (utilizando, por exemplo, os padrões de GAMMA *et al.* (1995)).

GAMMA *et al.* (1995) apresentam a seguinte diferenciação, considerando padrões de projeto e frameworks de aplicação:

- **Padrões de Projeto são mais abstratos que Frameworks.** Frameworks podem ser incorporados em código, enquanto que apenas exemplos de padrões podem ser incorporados em código;
- **Padrões de Projeto são elementos arquiteturais menores do que Frameworks.** Um típico framework contém vários padrões de projeto; e
- **Padrões de Projeto são menos especializados do que Frameworks.** Frameworks sempre pertencem a um domínio em particular, enquanto que os padrões de projeto poderiam, em princípio, ser usados em qualquer tipo de software. Mesmo padrões mais especializados, como por exemplo padrões para sistemas distribuídos, não determinariam a arquitetura da aplicação como acontece com os frameworks.

LAJOIE e KELLER (1994) consideram que os padrões de GAMMA *et al.* relatam micro-arquiteturas, definidas como estruturas de projeto orientado a objetos, que ocorrem repetidamente no desenvolvimento de frameworks de aplicação. Apresentam, então, uma classificação de granularidade de reutilização, em que aplicações seriam os

objetos de reutilização de maior granularidade, seguidos em ordem decrescente do framework de aplicação, da micro-arquitetura e, por fim, situados num mesmo nível, classes, objetos, métodos ou código.

Os autores apresentam estas diferenciações baseados em suas definições de frameworks e de padrões. Por exemplo, GAMMA *et al.* (1995), indicam os padrões como sendo de menor porte que os frameworks, por que seus padrões (aplicáveis à modelos de classes e objetos da fase de projeto) não compreendem mais do que uma dúzia de classes e objetos e os frameworks que consideram são os de aplicação, que normalmente apresentam um grande número de classes e objetos. No entanto, considerando uma definição mais ampla, padrões podem conter um grande número de classes e objetos e frameworks podem conter umas poucas classes e objetos. Portanto, uma diferenciação que considere definições mais abrangentes, elaborada a partir dos aspectos considerados acima, é apresentada a seguir:

- Um padrão pode encapsular um framework, apresentando-o como solução de algum problema. São exemplo disso, todos os padrões que apresentam um modelo de classes e objetos como solução. O framework pode ser tanto de pequeno porte (algumas poucas classes e objetos) como de grande porte (framework de aplicação) e sua especialização pode ser mais ou menos genérica. Apesar do objetivo dos padrões ser a universalidade, é concebível que um padrão seja específico a ponto de ser aplicável apenas em um domínio;
- Padrões são aplicáveis a frameworks, na medida em que apresentam melhoras em parte, ou na totalidade, da organização de classes e objetos com um determinado objetivo; e
- Um framework pode se prestar como o ponto de partida para a criação de um padrão. Por outro lado, um framework pode ser construído a partir de alguns padrões.

2.5. Formato de Padrões

O formato dos padrões especifica a estrutura de dados necessária para armazenar as informações apresentadas no padrão. É razoável afirmar, considerando-se a literatura sobre padrões, que existe um conjunto básico de informações importantes sobre as soluções que se deseja registrar, de forma a se tornarem úteis para consulta. Basicamente, os padrões devem apresentar os seguintes itens de informações:

- **Identificação:** Um nome, ou frase, descritivo, curto e familiar, normalmente sendo mais indicativo da solução do que do problema ou contexto;
- **Exemplo:** Um ou mais casos ilustrando o emprego do padrão;
- **Contexto:** Delineamento de situações em que os padrões são aplicáveis. Geralmente inclui informações de suporte, discussões sobre a necessidade de existência do padrão e evidências de sua generalidade;
- **Problema:** Uma descrição das forças e restrições relevantes e como interagem;
- **Solução:** Uma apresentação da solução do problema, descrita de forma a ser útil, já que esta é a parte reutilizável. Soluções podem se referenciar e se relacionar com outros padrões de maior ou menor nível;
- **Conseqüências:** Uma descrição das implicações da aplicação do padrão, como restrições e comprometimentos; e
- **Padrões Relacionados:** Outros padrões que têm alguma relação com este. Por exemplo, aqueles que podem ser usados em conjunto ou que são semelhantes.

Os trabalhos encontrados na literatura diferem em relação a estes sete itens, sobretudo devido ao contexto de aplicação dos padrões, mas também devido ao objetivo da linguagem de padrões (algumas linguagens de padrões apenas se preocupam em apresentar uma breve argumentação sobre os problemas e suas soluções). Baseado neste formato básico, é apresentada a seguir uma discussão sobre o que se considerou importante ressaltar dos formatos de alguns dos trabalhos encontrados na literatura.

O formato de Alexander se diferencia destes sete itens básicos por não considerar as conseqüências de aplicação (LEA, 1994).

Estes itens diferem dos quatro considerados básicos por GAMMA *et al.* (1995) (apresentados na seção 2.1) em relação à necessidade de exemplos e da indicação de outros padrões (embora não seja um item específico, o contexto é tratado no esquema básico de GAMMA *et al.* juntamente com o problema). A indicação de outros padrões não é essencial, mas considera-se que os exemplos sejam importantes por permitirem uma avaliação diferente do padrão que, por si só, pode esclarecer ou complementar as informações do padrão sobre contexto, conseqüências, problema e solução.

No formato de GAMMA *et al.* as informações figuram divididas em um número maior de itens. Por exemplo, a solução está apresentada nos itens estrutura, participantes

e colaborações. São apresentadas também informações bastante interessantes, para o contexto de aplicação dos seus padrões, sobre a implementação do padrão, além de fornecer código de amostra. São incluídas também a classificação do padrão com vistas a auxiliar sua identificação.

Os padrões de BUSCHMANN e MEUNIER seguem um formato semelhante ao dos padrões apresentados por GAMMA *et al.*, destacando-se por empregar cenários para a descrição do comportamento dinâmico do padrão (BUSCHMANN e MEUNIER, 1995).

Os padrões de MULARZ (1995) incluem uma análise do padrão sob o aspecto de desempenho, com o objetivo de disponibilizar ao usuário a possibilidade de obtenção de uma solução de compromisso entre funcionalidade e performance.

2.6. Classificação de Padrões

Apresentamos, nesta seção, esquemas de classificação de padrões propostos em alguns dos trabalhos abordados neste relatório técnico. Além de organizar os padrões, o esquema pode ser útil na aplicação dos padrões.

Segundo BUSCHMANN e MEUNIER (1995), um esquema de classificação busca focalizar as características principais dos padrões e, eventualmente, pode até ser usado para ajudar a organizar padrões nos sistemas em desenvolvimento. Observam ainda que o esquema de classificação pode facilitar a referência a padrões e grupos de padrões, bem como facilitar o seu aprendizado e a descoberta de novos (este é o objetivo explícito de GAMMA *et al.* para seu esquema de classificação).

A maioria dos esquemas de classificação enquadra os padrões segundo seus objetivos. No entanto, é comum encontrar esquemas que adotam mais de uma dimensão. Por exemplo, o esquema bi-dimensional de GAMMA *et al.* (vide tabela 1) classifica os padrões segundo seu propósito (criação, estruturação ou comportamento) e escopo de aplicação (aplicável primariamente a classes ou a objetos).

		PROPÓSITO		
		Criativo	**Estrutural**	**Comportamental**
ESCOPO	**Classe**	Método Fábrica	Adaptador (classe)	Interpretador Método Modelo
	Objeto	Construtor Fábrica Abstrata Protótipo Singular	Adaptador (objeto) Composto Decorador Fachada Peso-Mosca Ponte Procurador	Cadeia de Responsabilidade Comando Iterador Mediador Memorial Observador Estado Estratégia Visitador

Tabela 1 - Classificação de Padrões (GAMMA *et al.*, 1995)

Os esquemas de classificação, também, podem servir de base para a elaboração de um método de pesquisa de padrões em um repositório. Por exemplo, tendo-se em mãos os padrões de GAMMA *et al.*, e enfrentando-se um problema relativo à estruturação de objetos, pode-se procurar uma solução nos padrões classificados como "escopo = objeto" e "propósito = estrutural".

Dentre as características dos padrões usadas para sua classificação, pode-se citar (BUSCHMANN e MEUNIER, 1995, COAD *et al.*, 1995, COPLIEN, 1992, GAMMA *et al.*, 1995, JOHNSON, 1992, PREE, 1994, ZIMMER, 1995):

- Propósito ou objetivo;
- Restrição a um domínio de problema. Por exemplo, segundo ZIMMER alguns padrões de GAMMA *et al.* são restritos a um domínio de aplicação;
- Modo de estruturação da flexibilidade em relação ao domínio. Por exemplo, PREE observa que seus padrões tratam da flexibilidade em relação a domínios através de métodos variáveis ou de isolamento em classes;
- Funcionalidade. Por exemplo, COAD considera seus padrões como tendo uma das seguintes funcionalidades: padrão fundamental, de transação, de agregação, de planejamento e de interação.
- Origem. Por exemplo, PREE considera alguns padrões de COAD como derivados do framework "Model-View-Controller"; e
- Granularidade. Por exemplo, BUSCHMANN e MEUNIER avaliam o padrão segundo sua aplicação para estruturação de todo um sistema ou de sub-sistemas.

Destes trabalhos, o que se figura como o mais completo é o esquema de BUSCHMANN e MEUNIER, por isso é apresentado a seguir com detalhes.

O trabalho de BUSCHMANN e MEUNIER (1995) é o único trabalho em que uma classificação é apresentada, na qual é mencionado especificamente a preocupação de facilitar a busca de um padrão. No entanto, BUSCHMANN e MEUNIER ressaltam que seu esquema não tem a pretensão de permitir a determinação do padrão correto para uma dada situação, mas sim diminuir o escopo da busca de padrões adequados. Observam que seu esquema reflete apenas um ponto-de-vista. Além disso, observam que é possível que um padrão possa merecer duas diferentes classificações, sugerindo que, nestes casos, o padrão receba as diversas classificações adequadas.

Seu esquema de classificação prevê três categorias principais: granularidade, funcionalidade e princípios estruturais, subdivididas em subcategorias.

A categoria **Granularidade** é apresentada como sendo necessária pelo fato de problemas no desenvolvimento de software ocorrerem em diferentes níveis de abstração. Os três níveis de granularidade previstos são:

- **Framework de Arquitetura:** Visa a estruturação de sistemas de software. Fornece um conjunto de subsistemas pré-definidos, bem como a organização do relacionamento entre eles;
- **Padrão de Projeto:** Descreve um esquema básico para a estruturação de subsistemas e componentes de uma arquitetura de software, bem como os relacionamentos entre eles; e
- **Idioma:** Descreve como implementar componentes específicos de um padrão, sua funcionalidade ou seus relacionamentos com outros componentes. Em geral, são específicos para uma determinada linguagem de programação.

A categoria **Funcionalidade** é motivada pelo fato de todo padrão ser um modelo para implementação de uma funcionalidade específica. A funcionalidade prevista é:

- **Criação de Objetos:** Especificar como criar instâncias particulares de estruturas de objetos, recursivas ou com agregação;
- **Guia da Comunicação Entre Objetos:** Descrever como organizar a comunicação entre um conjunto de objetos que colaboram entre si, que podem ser externos ou remotos;

- **Acesso a Objetos:** Descrever como ter acesso a serviços e estado de objetos compartilhados ou remotos, de maneira segura, sem violar seu encapsulamento de estado e comportamento; e
- **Organização do Processamento de Tarefas Complexas:** Especificar como distribuir responsabilidades entre objetos, de forma a solucionar uma função ou tarefa mais complexa.

A categoria **Princípios Estruturais** é justificada pelo fato de padrões se basearem em princípios, que são:

- **Abstração:** Provê uma visão abstrata ou generalizada de uma específica (e em geral complexa) entidade ou tarefa de um sistema de software;
- **Encapsulamento:** Encapsula detalhes de um objeto específico, componente ou serviço, para remover dependências delas de seus clientes ou para protegê-las de acesso;
- **Separação de Responsabilidades:** Fatora responsabilidades específicas através de objetos ou componentes, para resolver uma tarefa específica ou certo serviço; e
- **Acoplamento e Coesão:** Remove ou relaxa os relacionamentos e dependências estruturais ou comunicacionais entre objetos fortemente acoplados.

3. EXEMPLOS DE PADRÕES

Apresentamos a seguir, resumidamente, algumas aplicações de padrões encontradas na literatura. Em particular, descreveremos os trabalhos que se referem a padrões de processos.

PREE (1994) apresenta padrões que esclarecem e permitem alternativas quanto à implementação de modelos em que se deseje separar as áreas específicas do domínio das que são genéricas. Seu objetivo é aperfeiçoar a construção de *frameworks de aplicação*. Como seus padrões podem ser aplicáveis a quaisquer conjuntos de classes, também o poderiam ser a padrões como os de GAMMA *et al.* e COAD. Por isto, os denomina *meta-padrões*.

COPLIEN (1995) observa que os padrões dão suporte a técnicas emergentes no projeto de software, fornecendo uma nova maneira de entender e criar programas de computador. O autor considera-os como uma das mais promissoras abordagens generativas, sendo particularmente adequados para a construção e evolução organizacional. Os *padrões organizacionais*, como COPLIEN os denomina, seriam, portanto, importantes para a implantação de novas técnicas de estruturação de programas, uma vez que estas precisam receber o suporte de técnicas gerenciais e estruturas organizacionais apropriadas.

COPLIEN ressalta que a análise de organizações sob a perspectiva de padrões não é novidade, mas sim empregá-los de uma forma generativa. Considerando que toda arquitetura é fundamentalmente preocupada com controle, sua proposta é a definição de uma arquitetura baseada na estruturação necessária para realizar o negócio, ao invés de uma arquitetura determinada pelo processo do negócio, como um meio de controlar indiretamente as pessoas na organização. Afirma que os padrões auxiliam na construção de novas organizações, além de ajudar a entendê-las, e que um bom conjunto de padrões organizacionais pode ajudar indiretamente a gerar o processo correto.

O trabalho de WHITENACK (1995), apresenta uma linguagem de padrões para dar orientação a analistas, desenvolvedores e gerentes de projeto envolvidos na definição de requisitos, para aplicações comerciais a serem desenvolvidas num ambiente orientado a objetos. Baseado em sua experiência, de que a análise de requisitos de um domínio de problema complexo, em geral, é insuficiente para começar um projeto bem sucedido,

bem como para evitar uma retomada das atividades de análise e especificação de requisitos, WHITENACK elaborou uma linguagem de padrões, que almeja os seguintes objetivos:

- Guiar analistas e desenvolvedores do produto na aplicação apropriada de um conjunto de técnicas e métodos, de modo que uma análise de requisitos e um entendimento do problema mais completos sejam atingidos;
- Fornecer uma estrutura básica para a definição e captura de requisitos antes do desenvolvimento, bem como para a avaliação, projeto, construção e teste de um produto de software apropriado; e
- Fornecer um meio de rastrear o projeto de um sistema até os objetivos de sistema e de negócio originais.

O trabalho de BUSCHMANN e MEUNIER, em (1995), visa a obtenção de um conjunto de padrões, denominado de um *sistema de padrões*, que permita o projeto da arquitetura de um sistema através da composição de padrões.

O trabalho de MULARZ (1995) explora a utilização de padrões no desenvolvimento baseado na integração de componentes. MULARZ pretende fornecer uma estratégia para este tipo de desenvolvimento que não apresente as deficiências encontradas nas estratégias correntes, tais como, informalidade e especificidade de projeto e de produto. O autor observa que a expansão do espaço de componentes acaba por impedir que a integração explícita do produto seja adequada para a determinação de esquemas de integração bem sucedidos. Os padrões são usados para a codificação de esquemas de integração, fornecendo soluções para os problemas que podem ocorrer nesta atividade.

3.1. Padrões de Processos

Padrões de Processo são aqueles que abordam temas relativos ao processo de desenvolvimento de software, como, por exemplo, organização de equipes, definição de atividades e elaboração de modelos do software. Padrões de processo podem ser úteis no estabelecimento de um novo processo de desenvolvimento ou na gerência do processo durante sua execução.

Em princípio, todos os padrões seriam interessantes, uma vez que, por definição, resolvem problemas recorrentes no desenvolvimento de software. No entanto, o interesse deste trabalho se estende sobre o planejamento e gerência de processos de desenvolvimento, sendo a abordagem enfocada nesta seção restrita aos padrões que tratam de temas destas áreas.

Uma das linguagem de padrões de processo encontradas na literatura trata apenas da organização e gerenciamento de equipes (HARRISON, 1996). Por exemplo, um dos padrões sugere algumas atitudes a serem tomadas pelo líder da equipe, de forma a fomentar a unidade da equipe. As demais são apresentadas a seguir.

Os padrões de COPLIEN (1995) e WHITENACK (1995) sugerem possíveis soluções para problemas da análise de sistemas, tais como: 1) formação da equipe; 2) atribuição de responsabilidades; 3) duração do desenvolvimento; 4) planejamento de testes; 5) gerência do projeto; 6) determinação do comportamento do sistema; 7) determinação da natureza essencial do domínio do problema; e 8) avaliação dos requisitos. Os padrões de COPLIEN ainda apresentam soluções para problemas tais como: 1) dificuldade de obter especialistas no assunto; 2) dificuldades da equipe decorrentes do grande número de mudanças; 3) dificuldade de manutenção da documentação; 4) interferências em excesso de origem externa à equipe; e 5) comunicação intra-equipe insatisfatória. Já os de WHITENACK incluem soluções aos seguintes problemas: 1) previsão do funcionamento do sistema no suporte a um processo de negócio; e 2) especificação de requisitos através de regras de negócio.

O trabalho de BUSCHMANN e MEUNIER (1995) apresentam um conjunto de padrões para o desenvolvimento de software. Compreende padrões em vários níveis de abstração, indo de paradigmas fundamentais, para a estruturação dos sistemas, até a implementação de decisões de projeto específicas. Por exemplo, padrões classificados como *Frameworks de Arquitetura,* abordam a estruturação de sistema em camadas ou usando uma arquitetura baseada no modelo Modelo-Visão-Controlador ("Model-View-Controller").

COCKBURN (1996) apresenta uma linguagem de padrões composta de padrões de princípio de projeto e de padrões de decisões de projetos derivadas dos princípios. COCKBURN sustenta que os princípios de projeto que norteiam o desenvolvimento precisam ser definidos, de maneira a serem seguidos durante todo o desenvolvimento, evitando o aumento da complexidade do software. Por exemplo, um princípio de projeto

apresentado é a separação entre a interface do usuário e o modelo da aplicação. Isto serviria para delimitar os impactos de mudanças na interface com o usuário, um item reconhecidamente sujeito a mudanças ao longo do ciclo de vida do software. Como decisões de projeto derivadas deste princípio de projeto, COCKBURN apresenta duas: 1) possibilidade de uso de ferramentas geradoras de interface do tipo "GUI"; e 2) facilidade de comando da aplicação a partir de outras plataformas.

CUNNINGHAM (1996) apresenta uma linguagem de padrões que aborda decisões a serem tomadas durante o desenvolvimento de software, por parte de um agente (um indivíduo, um grupo, um departamento ou a gerência), relativas a uma tarefa que envolve o produto, o desenvolvimento, a programação ou a operação. O objetivo da linguagem de padrões é preparar a organização para decisões cruciais e necessárias, de forma a diminuir o impacto destas decisões e manter o desenvolvimento num fluxo contínuo. Por exemplo, um padrão estabelece as ações a serem tomadas no sentido de esclarecer quais os objetivos específicos do produto. Um outro trata da elaboração de um organograma que não venha a ter conseqüências negativas na equipe.

Em (VASCONCELOS, 1997) é apresentada uma abordagem que emprega os padrões na extensão e organização de descrições de processos de desenvolvimento, no contexto de um ambiente de desenvolvimento de software, provendo uma infra-estrutura para definição e aplicação de padrões de processo. Desta forma, o conhecimento sobre processos é disponibilizado durante a execução de um projeto de desenvolvimento, facilitando sua reutilização pela gerência.

4. DESENVOLVIMENTO DE SOFTWARE UTILIZANDO PADRÕES

Nesta seção, são apresentadas observações encontradas na literatura, com relação ao desenvolvimento de software utilizando padrões. Estes trabalhos consideram a produção de software ocorrendo a partir de consultas a catálogos de padrões. Os trabalhos apresentados a seguir são os de ALFRED e MELLOR (1995), PREE (1994) e BUSCHMANN e MEUNIER (1995).

A produção dos catálogos de padrões é uma dificuldade inicial que, após resolvida, restará, ainda, a dificuldade da utilização destes catálogos. Como os padrões são por definição genéricos, pode ser difícil reconhecer em um modelo de domínio (cuja estrutura reflete conceitos e relações do domínio) um modelo genérico que reflete um comportamento abstrato. Uma compreensão aprofundada do que está sendo modelado no domínio pode ajudar, mas, certamente, é a experiência que vai facilitar. Por exemplo, o fato da descoberta de que o problema sendo modelado possui uma estrutura capturada por um padrão, permite que se faça uma análise crítica do modelo que estava sendo gerando e que se façam as correções necessárias para evitar os problemas já detectados e solucionados pelo padrão.

BUSCHMANN e MEUNIER (1995) apontam a necessidade de um método que guie o arquiteto de software na utilização de um sistema de padrões durante todo o projeto. Este guia precisa considerar esquemas de classificação que possibilitem a identificação de padrões apropriados para uma situação de projeto e a sua integração no software. Por exemplo, uma abordagem "top-down" pode não ser viável devido à existência de requisitos de nível mais baixo que afetem a estrutura com um todo.

Baseado no esquema de classificação tridimensional proposto (descrito na seção 2.6), que considera como categorias a *granularidade*, a *funcionalidade* e os *princípios estruturais*, os autores sugerem um procedimento para a seleção dos padrões candidatos. A seguir são listados os passos e os comentários apresentados por BUSCHMANN e MEUNIER (1995):

- **Determinar a *granularidade* requerida do padrão:** Os autores observam que normalmente trata-se de uma tarefa fácil;

- **Selecionar a *funcionalidade* desejada:** Consideram que deveria ser igualmente fácil, pois normalmente apenas uma categoria de funcionalidade é considerada para uma situação de projeto. No entanto, pode ser necessário combinar vários aspectos funcionais em uma única estrutura, o que acarretará a seleção de um único, ou de um conjunto de padrões; e
- **Determinar os *princípios estruturais* desejados:** Considerado o passo mais difícil. Será uma decisão de projeto, uma vez que normalmente mais de uma solução estrutural é possível. O desenvolvedor será forçado a pensar que propriedade estrutural é a mais útil e importante para a situação em consideração.

ALFRED e MELLOR (1995) fazem uma análise extremamente otimista (pode-se dizer, demasiadamente otimista) do desenvolvimento usando padrões, apresentando considerações sobre o ciclo de vida adequado, a equipe e as ferramentas necessárias. Os autores consideravam que no ano 2000, com o custo de produzir os modelos de projeto e de implementação, a partir de um modelo de análise completo, detalhado e testável, tendo se tornado suficientemente baixo, a análise de domínio vai tomar o lugar do que hoje se denomina desenvolvimento de software, sendo uma tarefa desempenhada por três tipos de profissionais:

- **Analista:** Um especialista em algum domínio de problema, capaz de desenvolver para um sistema um modelo independente de implementação, que seja completo, detalhado, executável e testável;
- **Projetista:** Um especialista numa área tecnológica ou domínio de problema abstrato, capaz de trabalhar com um catálogo de padrões, reconhecer a ocorrência deles numa análise de domínio e usar as restrições do problema para selecionar uma implementação apropriada para cada padrão; e
- **Integrador:** Um especialista na montagem de padrões específicos de implementação, bem como, no desenvolvimento, teste e implantação de um sistema.

ALFRED e MELLOR apresentam um modelo de ciclo de vida que consideravam factível de ser implantado pelas empresas até o ano 2000, que se caracteriza por ser uma variação do ciclo de vida em espiral, prevendo as fases de levantamento de requisitos, projeto, implementação, teste de unidades, integração e teste do sistema. Para a fase de

projeto, consideram a disponibilidade de uma base de conhecimento de projeto, que é um banco de dados de padrões para domínios abstratos, contendo uma ou mais implementações para cada padrão e as regras de seleção de uma implementação específica.

O modo como ALFRED e MELLOR apresentam a influência dos padrões no processo de desenvolvimento está descrita a seguir:

- **Fase de Levantamento de Requisitos:** Nesta fase, consideram fundamental a seleção de padrões de uso adequados;
- **Fase de Análise:** É papel do analista ter conhecimento sobre os padrões disponíveis na base de conhecimento de projeto (veja a seguir) e empregá-los na modelagem do domínio sendo analisado. Ao padrão selecionado, o analista adiciona atributos e comportamentos específicos do domínio; e
- **Fases de Projeto:** Para cada padrão do modelo de análise construído, o projetista aplica as regras da base de conhecimento. Escolhe uma implementação apropriada, consultando o conjunto de padrões de implementação, que irá adicionar ao modelo uma estrutura, atributos e comportamentos específicos da implementação. A escolha de um padrão de projeto vai depender dos padrões de uso selecionados na fase de levantamento de requisitos, bem como dos objetivos de desempenho e escala.

Sobre os padrões em si, ALFRED e MELLOR fazem as seguintes observações:

- **Padrões na Fase de Levantamento de Requisitos:** Os "use case" de JACOBSON *et al.* (1992) seriam uma excelente forma de descrever padrões de uso para sistemas interativos;
- **Padrões na Fases de Análise:** Padrões de análise ocorrem em uma grande variedade de domínios, podendo ter várias formas diferentes, mas com uma interface externa comum; e
- **Padrões na Fases de Projeto:** São estreitamente relacionados aos de análise. Um padrão de análise vai suscitar um ou mais padrões de projeto, um para cada maneira útil de representar os detalhes internos (Organização e comportamentos) dos componentes do padrão de análise. Diferentes padrões de projeto apresentam diferentes comprometimentos de espaço, tempo e flexibilidade.

ALFRED e MELLOR consideram que surgirão, tornando possível a automatização das fases de projeto e implementação, as seguintes ferramentas:

- **Geradores de aplicações** capazes de produzir todo o código de uma aplicação, baseados em notações lógicas de modelos de projeto;
- **Bases de dados de políticas e de padrões de projeto**, que serão a base para automatização da produção de um modelo de projeto, a partir do modelo de análise e de um conjunto de restrições de aplicação;
- **Ferramentas CASE** capazes de produzir modelos de análise completos, detalhados e testáveis.

PREE (1994) trata especificamente do desenvolvimento de frameworks. Assumindo que seus meta-padrões podem dar suporte efetivo a tal desenvolvimento e torná-lo mais eficiente. PREE propõe a abordagem orientada a "hot spots" (i.e., as áreas variáveis de um framework, que são sensíveis a mudanças de domínio), ilustrada na figura 3. PREE considera esta abordagem como um aperfeiçoamento necessário às metodologias existentes de análise e projeto orientados a objetos.

PREE afirma que os ciclos de vida mais conhecidos são muito pouco úteis ao desenvolvimento de frameworks, e cita o modelo de "cluster" de ciclo de vida (MEYER, 1990) como uma caracterização geral com alguma semelhança ao desenvolvimento de frameworks.

Observando o desenvolvimento de frameworks, PREE afirma que uma equipe de desenvolvimento de framework tem de estar preparada e disposta a radicalmente reprojetar e reimplementar componentes, se novas e/ou melhores abstrações forem encontradas. Mas a abordagem orientada a "hot spot" pode contribuir para que a abstração correta seja descoberta mais rápido, reduzindo assim o número de ciclos de reprojeto.

PREE enfatiza que um desenvolvimento bem sucedido de framework requer a identificação dos "hot spots" específicos do domínio, implementando de forma genérica os aspectos do framework que não podem ser antecipados para todas adaptações. Portanto, observa que os especialistas do domínio devem se perguntar: "Que aspectos diferem de uma aplicação para outra? Qual é o grau de flexibilidade desejado? Deve ser possível mudar o comportamento flexível em tempo de execução?"

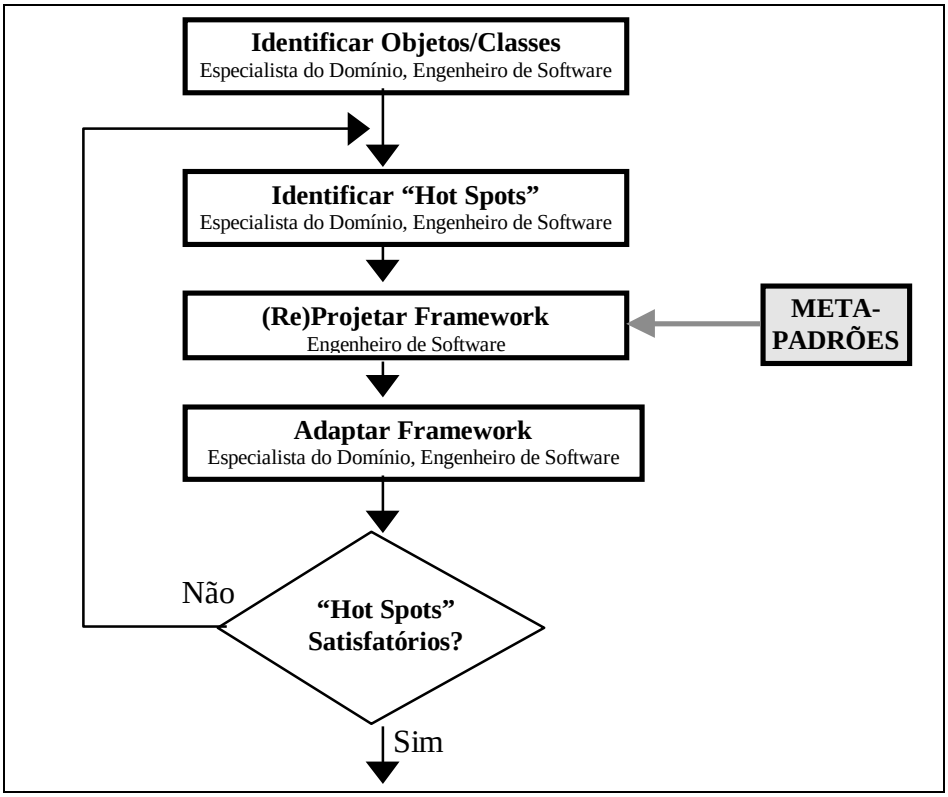

Figura 3 - Abordagem Orientada a "Hot Spot" (PREE, 1994)

Como ferramenta que auxilie a adaptação de frameworks e o desenvolvimento de novos frameworks utilizando meta-padrões, empregando a abordagem orientada a "hot spots", PREE sugere um tipo de ferramenta de hipertexto, o "browser" de meta-padrões, que teria a seguinte funcionalidade:

- Exibir o framework de forma gráfica ou textual (código fonte);
- Exibir num framework os meta-padrões utilizados;
- Exibir para um meta-padrão aplicado num framework, a correspondência entre os elementos do meta-padrão e do framework;
- Exibir todas as ligações entre um meta-padrão e frameworks onde é aplicado;
- Aplicar um meta-padrão a um determinado "hot spot" de um framework; e
- Aplicar, num determinado "hot spot" de um framework, o mesmo meta-padrão aplicado a um "hot spot" de um framework anteriormente selecionado.

BUSCHMANN e MEUNIER (1995) apontam problemas quanto a ausência de guias para a aplicação dos padrões e quanto a dificuldade de sua classificação. Considerando a construção do software através da combinação de padrões,

BUSCHMANN e MEUNIER advertem que não se trata de uma tarefa mecânica. É necessária a habilidade de um arquiteto de sistemas para compor os blocos de construção adequadamente, mas esta tarefa pode ser facilitada pelo emprego de orientações sobre a composibilidade de cada padrão e sobre a construção de uma estrutura a partir de padrões.

BUSCHMANN e MEUNIER apontam três questões a serem investigadas:

- **Completeza da linguagem de padrões:** Consideram importante discutir sobre a possibilidade / necessidade da linguagem de padrões ser completa;
- **Orientações sobre a construção de uma estrutura a partir de padrões:** Por exemplo, consideram que não é possível adotar uma abordagem "top-down", já que restrições de nível mais baixo podem afetar a estrutura como um todo; e
- **Esquema de classificação:** Consideram necessário aperfeiçoar o esquema de classificação que apresentam, verificando as categorias previstas, no sentido de corrigi-las e permitir um classificação mais exata. Ressaltam, ainda, que alguns desenvolvedores sugerem como alternativa aos esquemas de classificação, o uso de "mapas", contendo uma breve descrição dos padrões e suas conexões e ligações.

5. CONCLUSÃO

O formato dos padrões pode ser considerado como a definição de uma estrutura de dados para armazenar o conhecimento sobre problemas e suas soluções. Obviamente, a evolução da Ciência da Computação, como de qualquer atividade científica, ocorre em ciclos de: 1) levantar problemas; 2) propor soluções; e 3) aplicar as soluções, chegando num novo contexto (com novos problemas). Portanto, a tarefa de descrever soluções para problemas conhecidos, ou não, não tem nada de inovador. A inovação que os padrões apresentam está na forma sistemática em que são dispostas as descrições. Dessa forma, o conhecimento que contém fica valorizado, tendo sua aplicação facilitada. O objetivo de se dispor de manuais para o desenvolvimento de software, de maneira semelhante a que dispõe um engenheiro civil, pode ser difícil, no entanto, como se pode verificar pelo crescente número de trabalhos sobre padrões encontrados na literatura, haverá uma grande quantidade de conhecimento registrado de forma sistemática e orientado à reutilização. Portanto, torna-se necessário o desenvolvimento de ferramentas que auxiliem a aplicação dos padrões.

Tão importante quanto ter disponível o conhecimento sobre soluções de problema, é poder aplicá-las de maneira efetiva. Num ambiente de suporte automatizado ao desenvolvimento, uma ferramenta pode auxiliar na aplicação de um padrão. Os padrões devem levar em conta o contexto em que suas soluções serão aplicadas, de modo a permitir que a ferramenta possa realizar a implementação da solução.

Na medida em que possuem a preocupação básica de registrar de forma completa o conhecimento sobre problemas, os padrões podem ser úteis para representar o conhecimento de processos de desenvolvimento, considerando, por exemplo (como faz WHITENACK (1995)), como problemas, as atividades a serem realizadas de maneira correta. A utilização de padrões para este fim permite a criação de descrições de processos que incorporam a bagagem da organização em termos de sua experiência acumulada no desenvolvimento de software.

Por fim, deve-se fazer uma ressalva sobre o conhecimento capturado nos padrões. Embora a idéia dos padrões seja já de indiscutível utilidade, o valor do conhecimento que descrevem precisa ser examinado caso a caso. Os padrões tornaram-se um incentivo à divulgação de soluções para problemas específicos, e dentre a

diversidade de padrões encontrados na literatura é possível encontrar alguns em que as soluções apresentadas não representam o conhecimento de um especialista na área.

BIBLIOGRAFIA

C. **Alfred** e S.J. **Mellor;** "Observations on the Role of Patterns in Object-Oriented Software Development"; *Object Magazine*, v. 5, n. 2, pp. 60-65, Maio/1995.

G. **Arango** e R. **Prieto-Díaz;** "Domain Analysis Concepts and Research Directions"; In: R. Prieto_Diaz e G. Arango (eds), *Domain Analysis and Software Systems Modeling*; IEEE Computer Society Press Tutorial, 1991.

F. **Buschmann** e R. **Meunier;** "A System of Patterns"; In: J. Coplien e D. Scmidt (eds), *Pattern Languages of Program Design*; Addison-Wesley, Reading, MA, EUA, pp. 325-343, 1995.

B.G. **Cain** e J.O. **Coplien;** "A Role-Based Empirical Process Modelling Environment"; In: *Proceedings of the 2nd International Conference on Software Process*, Berlim, Alemanha, Fevereiro/1993.

P. **Coad, 1992**; "Object-Oriented Patterns"; *Communications of the ACM*, v. 35, n. 9, pp. 152-159, Setembro/1992.

P. **Coad**, D. **North** e M. **Mayfield**; *Object Models - Strategies, Patterns & Applications*; Yourdon Press Computing Series, Prentice-Hall, 1995.

A. **Cockburn**; "Prioritizing Forces in Software Design"; In: J.M. Vlissides, J.O. Coplien e N.L. Kerth **(**eds); *Pattern Languages of Program Design 2*; Addison-Wesley, Reading, MA, EUA; pp. 319-333, 1996.

J. **Coplien**; *Advanced C++ Programming Styles and Idioms*; Addison-Wesley, Reading, MA, EUA, 1992.

J. **Coplien**; "A Gerative Development-Process Pattern Language"; In: J. Coplien e D. Scmidt (eds), *Pattern Languages of Program Design*; Addison-Wesley, Reading, MA, EUA, pp. 183-237, 1995.

J. **Coplien** e D. **Scmidt** (eds); *Pattern Languages of Program Design*; Addison-Wesley, Reading, MA, EUA, 1995.

W. **Cunningham**; "EPISODES: A Pattern Language of Competitive Development"; In: J.M. Vlissides, J.O. Coplien e N.L. Kerth **(**eds); *Pattern Languages of Program Design 2*; Addison-Wesley, Reading, MA, EUA; pp. 371-388, 1996.

E. **Gamma**, R. **Helm**, R. **Johnson** e J. **Vlissides**; *Design Patterns - Elements of Reusable Object-Oriented Software*; Addison-Wesley, Professional Computing Series, Reading, MA, EUA, 1995.

N.B. **Harrison**; "Organizational Patterns for Teams"; In: J.M. Vlissides, J.O. Coplien e N.L. Kerth **(**eds); *Pattern Languages of Program Design 2*; Addison-Wesley; pp. 345-352, 1996.

I. **Jacobson**, M. **Christerson**, P. **Jonsson** e G. **Overgaard**; *Object-Oriented Software Engineering - A Use Case Driven Approach*; Addison-Wesley, Reading, MA, EUA, 1992.

R. **Johnson**; "Documenting Frameworks Using Patterns"; In: *Proceedings of the Conference on Object-Oriented Programming Systems, Languages, and*

Applications - OOPSLA'92, pp. 63-76, Vancouver, British Columbia, Canadá, 1992.

R. **Johnson**; "Why a Conference on Pattern Languages?"; *ACM SIGSOFT Software Engineering Notes*, v. 19, n. 1, pp. 50-52, Janeiro/1994.

R. **Johnson**; "A Report on PLoP'94"; *ROAD*, v. 1, n. 4, pp. 54-56, Novembro-Dezembro/1994.

R. **Lajoie** e R.K. **Keller**; "Design and Reuse in Object-Oriented Frameworks: Patterns, Contracts and Motifs in Concert"; In: *Proccedings of the 62nd Congress of the Association Canadienne Française pour l'Avancement des Sciences - ACFAS'94*, Montreal, Canadá, 1994.

D. **Lea**; "Christopher Alexander: An Introduction for Object-Oriented Designers"; *ACM SIGSOFT Software Engineering Notes*, v. 19, n. 1, pp. 39-46, Janeiro/1994.

D. **Lea**; *Patterns-Discussion FAQ*; Disponível por meio de ftp na Internet, Arquivo ca.ps, Servidor g.oswego.edu; E-mail do autor: dl@cs.oswego.edu; Versão de Março/1995.

D. **Mularz**; "Pattern-Based Integration Architectures"; In: J. Coplien e D. Scmidt (eds), *Pattern Languages of Program Design*; Addison-Wesley, Reading, MA, EUA, pp. 441-452, 1995.

W. **Pree**; *Design Patterns for Object-Oriented Software Development*; Addison-Wesley, 1994.

R.S. **Pressman**; *Software Engineering: A Practitioner's Approach*, McGraw-Hill International Editions, 3ª edição, 1992.

F.M. de **Vasconcelos** Jr.*; Reutilização de Processos de Desenvolvimento de Software Baseada em Padrões*; Tese de Mestrado, Programa de Engenharia de Sistemas e Computação, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil, 1997.

J.M. **Vlissides**, J.O. **Coplien** e N.L. **Kerth (**eds); *Pattern Languages of Program Design 2*; Addison-Wesley; 1996.

B. **Whitenack**; "RAPPeL: A Requirements-Analysis-Process Pattern Language for Object-Oriented Development"; In: J. Coplien e D. Scmidt (eds), *Pattern Languages of Program Design*; Addison-Wesley, Reading, MA, pp. 259-291, 1995.

W. **Zimmer**; "Relationships Between Design Patterns"; In: J. Coplien e D. Scmidt (eds), *Pattern Languages of Program Design*; Addison-Wesley, Reading, MA, pp. 345-364, 1995.