Um Estudo Comparativo do Tempo de Composição de um Framework Orientado a Aspectos de Persistência e de um Framework Orientado a Objetos de Persistência

Valter Vieira de Camargo[1,2], Erika Nina Höhn[1], José Carlos Maldonado[1]

[1]Instituto de Ciências Matemáticas e de Computação (ICMC)

Universidade de São Paulo (USP) – Avenida do Trabalhador São Carlense, 400

Cep 13.560-970 – São Carlos – SP

[2]Fundação de Ensino Eurípides Soares da Rocha – Univem

Avenida Hygino Muzzi Filho, 529 - Cep 17525-901 - Cx.Postal 2041 – Marília - SP

{valter, hohn, jcmaldon}@icmc.usp.br

Abstract. *This paper presents a quantitative study that investigated whether the use of aspect-oriented programming in implementation of persistence frameworks affects the composition time with a base-code. To conduct the study, two versions of the same persistence framework were used – one of them was implemented using aspect orientation, while the other was implemented using object orientation. Some students of a course participated in the study collecting data from the composition of the two frameworks with two applications. The results have shown that the composition time is smaller when using aspect-oriented programming.*

Resumo. *Este artigo apresenta um estudo quantitativo que investigou se o uso da orientação a aspectos na implementação de frameworks de persistência afeta o tempo de composição com um código-base. Para conduzir esse estudo, foram utilizadas duas versões do mesmo framework de persistência – uma implementada com orientação a aspectos, e outra implementada com orientação a objetos. Alunos de uma disciplina de pós-graduação participaram do estudo coletando os dados da composição dos dois frameworks com duas aplicações. Os resultados mostraram que o tempo de composição é menor quando se utiliza programação orientada a aspectos.*

1 – Introdução

Diferentemente dos frameworks orientados a objetos (FOO) de aplicação, cujo processo de reúso possui apenas a etapa de instanciação, os frameworks que tratam de apenas um interesse são reusados realizando-se duas etapas: instanciação e composição. A etapa de instanciação é o mesmo processo utilizado pelos FOO de aplicação, em que implementam-se ganchos do framework, escolhe-se alguma funcionalidade alternativa ou implementa-se uma nova. A atividade de composição, por sua vez, tem o objetivo de compor as variabilidades escolhidas na primeira na etapa de instanciação com o código-base, o que geralmente é feito com os mesmos mecanismos orientados a objetos utilizados na instanciação, causando entrelaçamento e espalhamento de código de diferentes interesses.

[1] Trabalho realizado com o apoio financeiro da CAPES.

Exemplos de frameworks orientados a objetos (FOOs) de persistência que se enquadram nessa categoria são: *Hibernate* (2006), *Cayenne* (2006) e *OJB* (2006).

Com o surgimento da programação orientada a aspectos (POA), e a possibilidade de encapsular interesses transversais (*crosscutting concerns*) de forma mais adequada [Kiczales et al., 1997], vários pesquisadores voltaram sua atenção para frameworks que tratam de um único interesse [Constantinides et al., 2000], [Pinto et al, 2002], [Rashid and Chitchyan, 2003], [Soares et al., 2002], [Vanhaute et al., 2001], [Couto et al., 2005], [Hanenberg et al., 2004]. Com os novos mecanismos disponíveis na POA, a composição desses frameworks com o código-base é facilitada e não apresenta os problemas de entrelaçamento e espalhamento existentes quando apenas orientação a objetos é usada. Os frameworks que tratam de um único interesse e que utilizam os novos mecanismos de composição da POA são denominados neste trabalho de Frameworks Transversais (*Crosscutting Frameworks*) (FTs) [Camargo e Masiero, 2005].

Vários trabalhos relatam que o uso da POA na implementação de frameworks de um único interesse resulta em benefícios de manutenção, reúso e legibilidade de código. Em particular, o interesse de persistência tem sido bastante pesquisado nesse sentido [Soares et al., 2002], [Couto et al., 2005], [Rashid and Chitchyan, 2003]. Contudo, não foram encontrados trabalhos cujo foco fosse averiguar se a POA também afeta o tempo de composição desses frameworks com o código-base. Esse é um fator importante porque a adoção de uma nova tecnologia, por exemplo, um framework, pode depender, além de outros fatores, da produtividade e eficiência que essa nova técnica traz para a equipe de desenvolvimento.

Dessa forma, este artigo apresenta um estudo quantitativo que comparou o tempo de composição de um framework OO e de um FT com um código-base. Para a realização do estudo foram utilizadas duas versões de um mesmo framework de persistência: uma orientada a objetos e uma orientada a aspectos. Esses frameworks foram compostos com duas aplicações similares e dados foram coletados.

Este artigo está organizado da seguinte forma: na seção 2, são apresentados os frameworks utilizados no estudo, bem como os passos necessários para sua composição com um código-base. Na Seção 3 o estudo de caso é descrito no formato proposto por Wohlin (2000). Na Seção 4 são mostrados os dados coletados e na Seção 5 a análise dos resultados. Na Seção 6 discutem-se possíveis ameaças aos resultados obtidos neste estudo e na Seção 7 encontram-se os trabalhos relacionados. Na Seção 8 estão as conclusões e os trabalhos futuros.

2. Frameworks Utilizados no Estudo

O estudo utilizou dois frameworks de persistência, que são, na realidade, duas versões – uma orientada a objetos e uma orientada a aspectos – de um mesmo framework. A versão orientada a objetos foi desenvolvida a partir da versão orientada a aspectos, e manteve a arquitetura, funcionalidades e características dessa versão, sendo modificada apenas a técnica utilizada para a composição com o código-base.

Como o FT de persistência havia sido desenvolvido com a preocupação de utilizar aspectos apenas onde fosse necessário, a criação de uma versão OO foi facilitada. Por exemplo, no FT de persistência, uma das tarefas da etapa de composição é fazer com que as classes persistentes da aplicação estendam uma interface do FT chamada `PersistentRoot`, o que é feito com declarações inter-tipo da linguagem AspectJ. No caso do framework de persistência OO, as classes persistentes da aplicação também devem estender a mesma interface, porém isso é feito introduzindo-se declarações

"`extends PersistentRoot`" em cada uma das classes. Outro exemplo é a composição da conexão, que no FT é feita concretizando-se dois conjuntos de junção (*pointcuts*) por meio da criação de um aspecto concreto. Já no framework OO, essa composição é feita por meio da invocação de métodos para abrir e fechar a conexão em vários.

Como comentado anteriormente, o processo de reúso de um framework que trata de um único interesse é feito em duas etapas: instanciação e composição. A etapa de instanciação para as duas versões do frameworks de persistência é idêntica, e consiste em escolher o tipo de conexão com o banco de dados. Como o objetivo do estudo foi comparar apenas a etapa de composição, os participantes do estudo receberam os frameworks com a instanciação já realizada, bastando apenas realizar a etapa de composição do framework com o código-base.

O framework de persistência utilizado no estudo, independentemente da versão (OO ou OA), possui duas partes distintas. A primeira parte corresponde ao que é chamado de "comportamento da persistência", que é um conjunto de operações básicas de persistência que devem ser adicionadas nas classes persistentes da aplicação. A segunda parte é o tratamento da conexão com o banco de dados, o qual consiste em determinar pontos da aplicação em que a conexão deve ser aberta e fechada. A etapa de composição desse framework possui três passos, sendo que os dois primeiros correspondem ao acoplamento dessas duas partes com o código base, e o último aos testes de acoplamento. Embora essas duas partes formem a persistência como um todo, estão sendo tratadas de forma separada porque a composição com o código-base utiliza técnicas diferentes de programação, tanto na versão OO quanto na OA. Coletando separadamente os dados foi possível averiguar qual das partes tem mais impacto no tempo de composição total.

2.1 – Passos da Etapa de Composição dos Frameworks

Os passos da etapa de composição são: 1) Acoplar Comportamento de Persistência; 2) Acoplar Conexão e 3) Testar Acoplamento. Esses são os passos que os participantes do estudo deveriam realizar para coletar dados. Para os dois primeiros passos, os participantes deveriam registrar o tempo de realização do passo, a quantidade de linhas de código escritas, a quantidade de locais modificados e a quantidade de unidades criadas, isto é, classes, pacotes e aspectos. Para o último passo apenas o tempo deveria ser registrado. Na Tabela 1 são mostrados os dados que deveriam ser coletados e a respectiva sigla de cada um.

Tabela 1 – Significado das Siglas dos Dados Coletados

Legenda dos Dados Coletados	
Dados para Coletar	**Siglas**
Tempo de Acoplamento do Comportamento de Persistência **(Passo 1)**	TACP
Tempo de Acoplamento da Conexão **(Passo 2)**	TAC
Tempo de Teste do Acoplamento **(Passo 3)**	TTeste
Quantidade de Linhas de Código Escritas	LC
Quantidade de Unidades (classes, aspectos, pacotes) Criadas	UC
Número de Locais modificados	LM

O primeiro passo da etapa de composição do FT consiste em criar um aspecto que estende o aspecto abstrato `OORelationalMapping` do framework e que declare, por meio de declarações inter-tipo, todas as classes persistentes da aplicação que devem implementar a interface `PersistentRoot`, como é mostrado na Figura 1. Para cada classe persistente de aplicação, por exemplo: `Endereco` e `Funcionário`, uma linha deve ser escrita informando que essa classe implementa a interface `PersistentRoot`.

Já no caso do framework OO de persistência, a realização do primeiro passo consiste em inserir a declaração "`extends PersistentRoot`" e a declaração de importação "`import persistence.PersistentRoot`", em cada classe persistente de aplicação, assim como está sendo mostrado na Figura 2.

```
public aspect MyOORelationalMapping extends OORelationalMapping {
       declare parents : Departamento    implements PersistentRoot;
       declare parents : Endereco        implements PersistentRoot;
       declare parents:  Funcionario     implements PersistentRoot; }
```

Figura 1 – Acoplamento do Comportamento de Persistência – AO

```
import persistence.PersistentRoot;
       public class [ClassName] extends PersistentRoot { … }
```

Figura 2 – Acoplamento do Comportamento de Persistência – OO

O segundo passo da etapa de composição consiste em estabelecer os locais do código onde a conexão deve ser aberta e fechada. Geralmente isso deve ser feito em programas (código-cliente) que usam objetos das classes persistentes definidas no passo anterior. No caso do FT, um aspecto concreto deve ser criado para estender o aspecto abstrato `ConnectionComposition` do FT, como está sendo mostrado na Figura 3. O aspecto abstrato `ConnectionComposition` possui dois conjuntos de junção abstratos: `openConnection()` e `closeConnection()` que devem ser concretizados. O conjunto de junção `openConnection()` deve ser concretizado de forma a entrecortar pontos da aplicação em que a conexão deve ser aberta, e o `closeConnection()` deve entrecortar pontos em que a conexão deve ser fechada. Além disso, também deve-se implementar o método gancho `getNameOfConnectionVariabilitiesClass()`. Esse método deve ser implementado de forma a retornar o nome da classe que foi criada na etapa de instanciação. Como a instanciação já havia sido realizada, o nome dessa classe também foi informado aos participantes.

```
public aspect myConnectionCompositionRules extends ConnectionComposition {

 pointcut openConnection(): execution (public static void SomeClass.main(..));
 pointcut closeConnection(): execution(public static void SomeClass.main(..));

  public String getNameOfConnectionVariabilitiesClass(){
             return "instantiation.myConnectionVariabilities";
   }}
```

Figura 3 – Acoplamento da Conexão – AO

Para o framework OO, a realização do segundo passo consiste em instanciar um objeto da classe `MyConnectionVariabilities` e chamar o método `ConnectDB()` em todos os pontos do código-base onde a conexão deve ser aberta e fechada. O estabelecimento da conexão deve ser feito logo no início dos programas e o encerramento deve ser feito no final. Na Figura 4 são mostradas em negrito as linhas de código que devem ser inseridas para que a composição possa ser realizada com um programa.

A etapa de teste foi idêntica para ambos os frameworks. Consistia em executar determinados programas e verificar se a saída correspondia à saída esperada. Por exemplo, uma das aplicações usadas no estudo possui um programa chamado `RegisterOfEvents`, cujo objetivo é armazenar em várias tabelas do banco de dados eventos que os

funcionários podem gerar durante o mês, por exemplo faltas, horas extras e atrasos. Se a execução desse programa não armazenar todos os dados esperados, a etapa de composição não pode ser considerada concluída e o erro deve ser rastreado e corrigido. Optou-se por não fornecer interfaces para as aplicações para que o tempo dos testes não fosse influenciado. O fornecimento dos programas fez com que todos realizassem os mesmos testes em tempos bastante próximos.

```
public class admissaoDeAssalariado {
  public static void main(String[] args) {
   MyConnectionVariabilities con = new MyConnectionVariabilities();
   con.ConnectDB();
      // Todo o comportamento do programa
   con.DisconnectDB();
   }}
```

Figura 4 – Acoplamento da Conexão - OO

3 – Descrição do Estudo Comparativo

Esta seção apresenta a definição, o planejamento e a execução do estudo de caso no formato proposto por Wohlin (2000). A análise dos resultados é mostrada na Seção 6.

3.1 - Definição do estudo

O objetivo foi **Analisar** o tempo de composição de um FT de persistência com uma aplicação e de um FOO de persistência com uma aplicação **para o propósito de** avaliação **com respeito à** eficiência **do ponto de vista de** desenvolvedores de software n**o contexto de** estudantes de pós-graduação em Ciências da Computação.

3.2 - Planejamento do estudo

a) Seleção do contexto. O estudo foi realizado com estudantes de pós-graduação em Ciência da Computação, no contexto da disciplina de Tópicos Avançados em Engenharia de Software, utilizando aplicações de domínio genérico (Loja de CDs e Oficina Eletrônica).

b) Formulação de hipóteses. Foram elaborados três tipos de hipóteses para o estudo: as hipóteses para o efeito do uso do Framework de Persistência no resultado podem ser vistas na Tabela 2; as hipóteses para o efeito do uso das Aplicações no Resultado encontram-se na Tabela 3, e as hipóteses para o efeito do grupo ou efeito da Interação Aplicação X Framework estão na Tabela 4.

c) Seleção de variáveis. As variáveis independentes são todas aquelas que são manipuladas e controladas durante o estudo. No estudo realizado essas variáveis são os Frameworks de persistência (OO e OA) e as aplicações (Loja de CDs e de Oficina Eletrônica).

As variáveis dependentes são aquelas que estão sob análise. Devem-se observar suas variações com base nas mudanças feitas nas variáveis independentes. No caso do estudo realizado a variável dependente é o tempo de composição.

d) Seleção dos participantes. Os participantes do estudo foram selecionados por meio de amostragem não-probabilística por conveniência. Eram alunos de pós-graduação da disciplina de Tópicos Avançados em Engenharia de Software, ministrada no ICMC/USP.

Tabela 2 – Hipóteses para o Efeito do Framework de Persistência

H_0:	Não há diferença entre desenvolvedores utilizando Framework de Persistência OA e desenvolvedores utilizando Framework de Persistência OO com respeito à eficiência individual.
Ha:	Há diferença entre desenvolvedores utilizando Framework de Persistência OA e desenvolvedores utilizando Framework de Persistência OO com respeito à eficiência individual.

Tabela 3 – Hipóteses para o Efeito do Uso das Aplicações no Resultado

H_0:	Não há diferença entre desenvolvedores fazendo o acoplamento na Aplicação Loja de CDs e desenvolvedores fazendo o acoplamento na Aplicação Oficina Eletrônica com respeito à eficiência individual.
Ha:	Há diferença entre desenvolvedores fazendo o acoplamento na Aplicação Loja de CDs e desenvolvedores fazendo o acoplamento na Aplicação Oficina Eletrônica com respeito à eficiência individual.

Tabela 4 – Hipóteses para o Efeito do Grupo ou Efeito da Interação

H_0:	Não há diferença entre o grupo que fez o acoplamento de um FOA com uma aplicação e o grupo que fez o acoplamento de um FOO em relação a eficiência.
Ha:	Há diferença entre o grupo que fez o acoplamento de um FOA com uma aplicação e o grupo que fez o acoplamento de um FOO em relação a eficiência.

e) Projeto do estudo realizado. O estudo foi planejado em bloco e balanceado para assegurar que os tratamentos tivessem igual número de participantes e para possibilitar a comparação dos efeitos das variáveis independentes, como pode ser visto na Figura 5. Na segunda fase do estudo, os 4 participantes mais experientes em OO e OA e nas linguagens Java e AspectJ executaram novamente as etapas do projeto experimental anterior, sendo invertida as combinações dos frameworks com as aplicações (Figura 5).

			Aplicações	
			Loja de CDs	**Oficina Eletrônica**
Framework de Persistência	1ª. Fase	**Treinamento**: exemplo de Composição do Framework de Persistência OO e OA em uma terceira aplicação.		
		Framework OO	Grupo 1	Grupo 2
		Framework OA	Grupo 2	Grupo 1
	2ª Fase	**Framework OO**	P4, P12	P1, P11
		Framework OA	P1, P11	P4, P12

Figura 5 – Projeto Experimental

f) Tipo de projeto: dois fatores com dois tratamentos (2*2 fatorial)

A distribuição dos participantes nos grupos foi realizada visando a colocar a mesma quantidade de participantes experientes nos dois grupos. A experiência dos participantes foi avaliada pelo questionário de caracterização dos participantes – documento utilizado para caracterizar experiência profissional e experiência nos assuntos relacionados com o estudo (conhecimento em OO e OA e das linguagens JAVA e AspectJ). O gráfico mostrado na Figura 6 (a) mostra a experiência dos participantes em Java, AspectJ e POA, e o gráfico (b) mostra a experiência com programação em Java e AspectJ. Os participantes P1, P2, P7 e P8 foram considerados experientes em POA.

g) Instrumentação. Os participantes deveriam contabilizar o tempo gasto para realizar a composição utilizando cada um dos frameworks.

Os documentos utilizados no estudo foram: formulário de caracterização dos participantes, para obtenção da experiência profissional e nos assuntos relacionados diretamente ao estudo; Formulário de registro, para preenchimento de todas as

informações durante a execução do estudo; Roteiro de execução, roteiro com todos os passos a serem seguidos para execução do estudo.

Os frameworks utilizados foram um de persistência orientado a objetos e um de persistência orientado a aspectos. E as duas aplicações foram uma para loja de CDs e uma para Oficina Eletrônica. A aplicação de loja de CDs possui 6 classes e 13 programas. O sistema de oficina eletrônica, possui 7 classes e 12 programas. As aplicações possuem complexidade similar e aproximadamente o mesmo número de linhas de código. Assim como os frameworks, essas aplicações foram desenvolvidas no meio acadêmico.

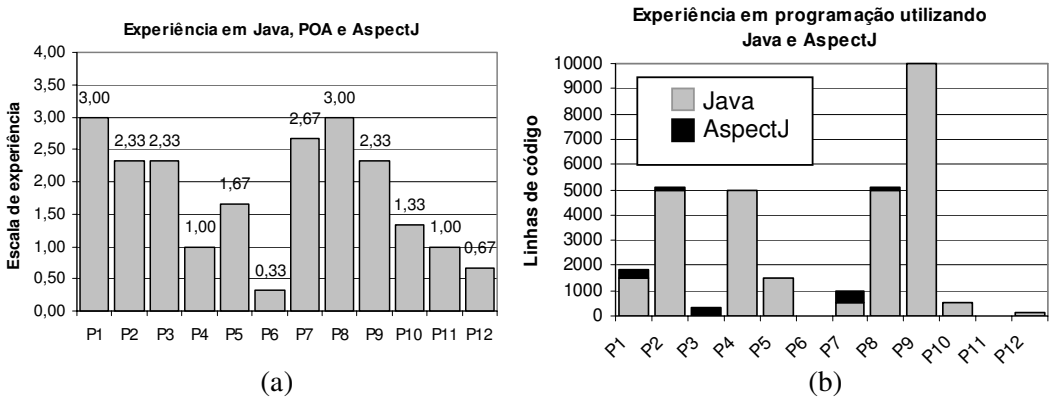

Figura 6 – (a) Experiência dos participantes em POA, Java e AspectJ; (b) Experiência dos participantes nas linguagens JAVA e AspectJ (linhas de código)

Quanto ao treinamento, foi realizado um pequeno treinamento visando a uniformizar o conhecimento sobre programação orientada a aspectos e de Frameworks (OO e OA). Também fez parte do treinamento a utilização de uma terceira aplicação, com a qual deveriam ser feitos os acoplamentos utilizando os frameworks OO e OA e todos os formulários que seriam utilizados no estudo posteriormente, para que o aprendizado sobre como fazer uma composição utilizando os frameworks e como preencher os formulários, não interferisse no tempo de acoplamento durante o estudo.

O objetivo da primeira fase foi realizar uma avaliação pura do tempo de composição e por isso, os pontos de acoplamento do código-base foram informados. Já na segunda fase, o objetivo foi avaliar, além do tempo que a técnica de programação causa, se a existência de uma nova dificuldade também interfere nos resultados.

3.3 - Execução. Os participantes executaram as atividades planejadas seguindo o projeto experimental. Primeiro treinaram a composição de um framework de persistência OO e depois de um FT em uma aplicação desenvolvida para exemplo.

Na primeira etapa da execução, o Grupo 1 utilizou o Framework de Persistência OO com a aplicação Loja de CDs, enquanto que o Grupo 2 utilizou o mesmo Framework com a aplicação Oficina Eletrônica. Na segunda etapa, os grupos utilizaram o Framework de Persistência OA, sendo que trocaram as aplicações, o grupo 1 utilizou a aplicação Oficina Eletrônica e o Grupo 2 utilizou a aplicação Loja de CDs.

4 – Dados Coletados

Esta seção mostra os dados coletados das duas fases do estudo. Nas Tabelas 5 e 6 são mostrados os dados da primeira fase do estudo, em que havia tanto participantes

experientes quanto novatos, e na Tabela 7, são mostrados os dados da segunda fase, em que havia apenas participantes experientes. Os significados das siglas dos dados coletados podem ser visto na Tabela 1.

Tabela 5 – Primeira Etapa da Primeira Fase – Composição com o Framework Orientado a Objetos

Primeira Etapa: Composição com Framework Orientado a Objetos de Persistência														
	Aplicação Loja de CDs – Grupo 1							Aplicação Abaco – Grupo 2						
	P1	P2	P3	P4	P5	P6	**média**	P7	P8	P9	P10	P11	P12	**média**
Dados														
1- TACP	01:55	10:00	02:11	02:07	01:58	04:00	**03:42**	05:00	01:58	01:40	03:05	04:00	06:00	**03:37**
1.1 - LC	10	10	10	10	10	10		16	16	16	16	16	16	
1.2 - UC	0	0	0	0	0	0		0	0	0	0	0	0	
1.3 - LM	5	5	5	5	5	5		8	8	8	8	8	8	
2- TAC	07:45	05:00	05:14	07:50	05:17	15:00	**07:41**	06:00	04:00	05:51	08:10	05:30	08:00	**06:15**
2.1 - LC	48	48	48	48	48	48		44	44	44	44	44	44	
2.2 - UC	0	0	0	0	0	0		0	0	0	0	0	0	
2.3 - LM	12	12	12	12	12	12		11	11	11	11	11	11	
3-TTeste	09:20	06:00	03:26	04:04	03:50	05:00	**05:17**	08:00	00:56	07:09	06:36	06:35	12:00	**06:53**
Soma dos Tempos	19:00	21:00	10:51	14:01	15:57	24:00	**16:40**	19:00	06:54	14:40	17:51	16:05	26:00	**16:45**

Tabela 6 – Segunda Etapa da Primeira Fase – Composição com o FT de Persistência

Segunda Etapa: Composição com FT de Persistência														
	Aplicação Abaco – Grupo 1							Aplicação Loja de CDs – Grupo 2						
	P1	P2	P3	P4	P5	P6	**média**	P7	P8	P9	P10	P11	P12	**média**
Dados														
1- TACP	05:45	04:00	03:10	03:00	03:10	02:02	**03:31**	04:00	02:22	03:33	02:58	03:12	10:00	**04:21**
1.1-LC	13	13	10	13	12	12		10	9	8	9	9	8	
1.2-UC	2	2	2	2	2	2		2	2	2	2	2	2	
1.3- LM	0	0	1	0	0	0		0	0	0	0	0	0	
2- TAC	02:30	01:00	03:35	01:54	02:30	04:00	**02:35**	02:00	00:42	03:40	02:57	02:32	03:00	**02:29**
2.1 - LC	10	12	8	12	10	7		10	8	9	12	10	7	
2.2 – UC	1	1	1	1	1	1		1	1	1	1	1	1	
2.3 – LM	0	0	1	0	0	0		0	0	0	0	0	0	
3-TTeste	08:50	07:00	04:30	07:41	03:56	07:00	**06:29**	06:00	00:40	05:55	05:30	03:55	03:00	**04:10**
Soma dos tempos	17:05	12:00	11:15	12:35	09:36	13:02	**12:35**	12:00	03:44	13:08	11:25	09:39	16:00	**10:59**

Os dados foram coletados separadamente para cada passo da etapa de composição do framework, isto é, foram coletados tempo, número de linhas de código escritas, número de locais modificados e número de unidades criadas para cada um dos três passos do processo de composição, a menos do passo de teste, em que foi contabilizado apenas o tempo. Os dados mostrados nas Tabelas 5, 6 e 7 estão agrupados pelos três passos da composição. No primeiro passo, representado pelo dado "1 - TACP" (veja Tabela 4), os dados "1.1 – LC", "1.2 – UC" e "1.3 – LM", que se encontram logo abaixo desse dado e que estão numerados como subseções, representam os dados coletados durante a realização desse primeiro passo. O mesmo ocorre com o segundo passo "Acoplar Conexão", representado pelo dado "2 – TAC". A letra "T" significa "tempo".

A coluna "média" mostra a média de tempo que os participantes gastaram na realização de cada um dos três passos e a última linha mostra a soma dos tempos de cada

participante e também a soma dessas médias, o que representa o tempo médio de composição do framework com a aplicação.

Note-se na Tabela 5 que a média de tempo de composição do FOO com a aplicação de loja de CDs (16:40) foi muito similar à média de tempo da composição desse mesmo framework com a aplicação Abaco (16:45), o que evidencia que as aplicações são similares em tamanho e funcionalidades. Mesmo no caso do FT, essa mesma similaridade é mantida, como pode ser visto na Tabela 6, a composição com a aplicação de loja de CDs foi realizada em 10:59 e com a aplicação Abaco em 12:35. Não se pode afirmar que a hipótese Ha mostrada na Tabela 3 pode ser refutada, porém há mais indícios que sim do que não.

Tabela 7 – Primeira e Segunda Etapas da Segunda Fase do Estudo

Primeira Etapa – Framework Orientado a Objetos								**Segunda Etapa – Framework Transversal**					
	Abaco – Grupo1			**Loja de CDs – Grupo 2**				**Loja de CDs – Grupo 1**			**Abaco – Grupo 2**		
	P1	**P2**	**média**	**P7**	**P8**	**Média**		**P1**	**P2**	**Media**	**P7**	**P8**	**Média**
Dado													
1- TACP	07:42	07:00	07:21	06:35	03:10	04:53		03:20	03:00	03:10	04:00	04:25	**04:13**
1.2-LC	16	16		10	10			10	11		9	14	
1.3-UC	0	0		2	0			1	1		1	1	
1.4- LM	8	8		5	5			0	0		0	0	
2-TAC	06:30	07:00	06:45	06:15	06:24	06:19		03:15	02:00	02:38	05:00	00:45	**02:53**
2.1 - LC	44	44		36	48			12	12		9	9	
2.2 – UC	0	0		0	0			1	1		1	1	
2.3 – LM	11	11		12	12			0	0		0	0	
Teste	06:05	07:00	06:32	05:00	04:25	04:43		04:18	07:00	05:39	04:00	04:10	**04:05**
Soma dos Tempos	20:17	21:00	**20:39**	17:50	13:59	**15:54**		10:53	12:00	**11:27**	13:00	09:20	**11:10**

Sempre que problemas ocorriam durante a coleta dos dados, os participantes eram instruídos a relatar o problema. Por exemplo, durante a realização da primeira etapa da primeira fase, o participante P2 realizou o acoplamento do comportamento de persistência (TACP) em 10 minutos, um valor bem acima da média dos outros participantes. Contudo, ele relatou que teve problemas com o ambiente Eclipse e que demorou cerca de 7,5 minutos para corrigir o problema. Note-se que se esse valor fosse descontado do tempo relatado pelo participante, ele estaria na média. Isso não foi feito porque problemas desse tipo são comuns durante qualquer desenvolvimento.

5 - Análise dos Dados

5.1 – Primeira Fase do Estudo

Os dados coletados na primeira fase do estudo mostraram que a composição utilizando o FT foi mais eficiente do que com o framework OO para 11 dos 12 participantes, como pode ser observado na Figura 7. Nessa figura, cada par de colunas representa os tempos de composição de um determinado participante. As colunas brancas representam o tempo gasto na composição com o framework OO e as colunas cinza com o FT. As etiquetas sobrepostas sobre cada par de colunas representam a diferença percentual entre os tempos. Por exemplo, o participante P2 realizou a composição com o FT 42,8% mais rápido do que com o framework OO. Note-se que apenas o participante P3 gastou mais tempo com o FT do que com o framework OO, porém foi uma diferença muito pequena: 21 segundos.

Analisando-se os formulários de coleta de dados desse participante não foi constatado nenhum problema.

Um fato interessante foi que os participantes com menos experiência em linguagens de programação obtiveram um nível de eficiência elevado com o FT. Por exemplo, o participante P6 fez a composição com o FT 31,4% mais rápido do que com o FOO, e o participante P12 obteve um índice ainda melhor, 38,4%.

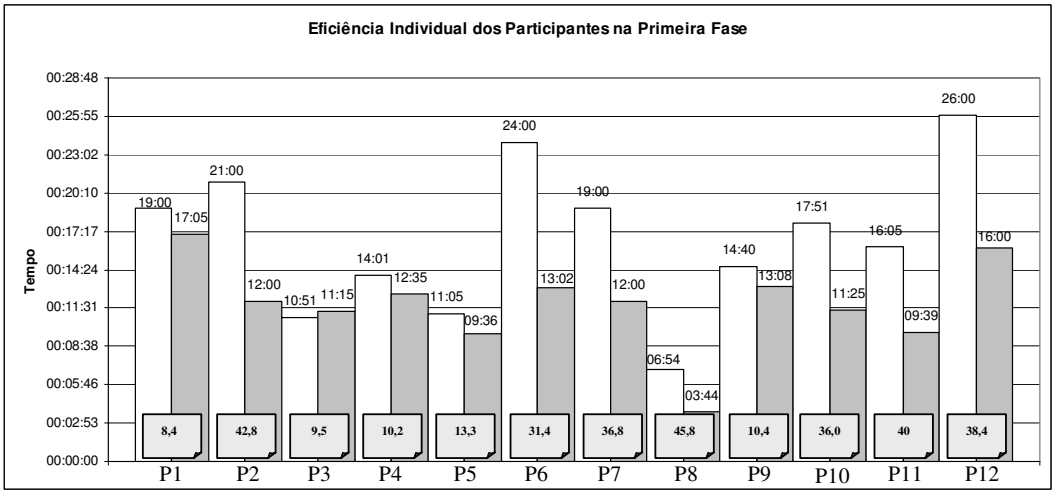

Figura 7 – Diferenças Percentuais entre os Tempos de Composição (Primeira Fase)

Desconsiderando os participantes experientes (P1, P2, P7 e P8), a média de tempo de composição dos novatos com o FT foi 23,65% menor do que usando o framework OO. Mesmo se os dados dos participantes P6 e P12, que são inexperientes em programação, forem desconsiderados, ainda assim os demais novatos foram 21,27% mais eficientes com o FT do que com o framework OO.

Os participantes experientes também foram mais eficientes com o FT do que com o framework OO. A média do tempo de composição foi 8,33% mais rápida com o FT do que os novatos – enquanto os experientes realizaram a composição com o FT 31,98% em média mais rápido do que com o framework OO, a média dos novatos foi de 23,65%.

Apenas o P1 obteve uma diferença percentual pequena entre os tempos, como pode ser observado na Figura 7. Porém, analisando-se os formulários desse participante, pôde-se constatar que ele obteve problemas com o ambiente durante a composição com o FT, o que elevou o tempo de composição com esse framework. Isso também pode ser constatado observando que o tempo de composição usando o FT desse participante foi superior ao de todos os outros participantes (17:05).

Sem considerar diferença entre experientes e novatos, o tempo médio de composição do FT com a aplicação de loja de CDs foi de 10:59 (Tabela 6), um valor 34,1% menor do que o tempo médio de composição dessa mesma aplicação com o framework OO, que foi de 16:40 (Tabela 5). Da mesma forma, o tempo médio de composição do FT com a aplicação Abaco foi de 12:35 (Tabela 6), um valor 24,87% menor do que com o framework OO, que foi de 16:45 (Tabela 5).

Assim, a média de tempo utilizada pelos dois grupos para a composição do FT foi de 11:47, enquanto que utilizando o framework OO foi de 16:42. O FT foi 29,3% mais rápido em média do que com o framework OO nessa primeira fase do estudo.

5.2 – Segunda Fase do Estudo

Os dados obtidos na segunda fase do estudo podem ser vistos no gráfico da Figura 8. Assim como na Figura 7, as notas sobrepostas sobre cada par de colunas representam a diferença percentual do tempo de composição usando cada um dos frameworks. Nesta fase, que só envolveu participantes com experiência em POA, o tempo de composição com o FT foi menor do que com o framework OO para todos os participantes. A eficiência média também foi melhorada para os participantes experientes, enquanto que na primeira fase a composição com o FT foi 31,98% mais rápida, nessa segunda fase foi de 37,37%. Uma diferença provavelmente ocasionada pelo conhecimento do framework.

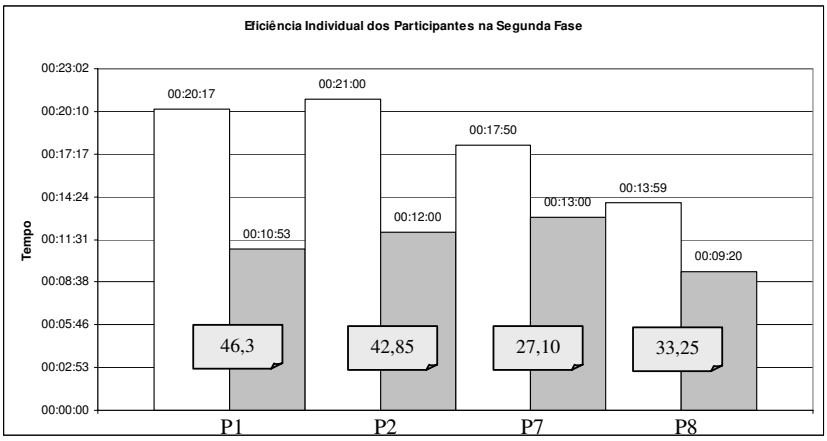

Figura 8 - Diferenças Percentuais entre os Tempos de Composição (Segunda Fase)

Considerando-se as médias de tempo, a composição da aplicação Abaco com o framework OO foi realizada em 20:39, enquanto que com o FT o tempo foi de 11:10, uma diferença de 54,0% a favor da orientação a aspectos. Já, com a aplicação de Loja de CDs a diferença foi bem menor, mas ainda assim favoreceu o FT, a composição com o framework OO foi realizada em 15:54, enquanto que com o FT foi realizada em 11:27, uma diferença de 27,98%.

A média de tempo utilizada pelos dois grupos de experientes para a composição do FT foi de 11:18 nesta segunda fase, enquanto que na primeira foi de 11:12, uma diferença insignificante. Embora também pequena, a diferença com o framework OO foi maior entre as duas fases. Enquanto que na primeira fase a composição com o framework OO foi realizada pelos experientes em 16:28 em média, na segunda foi realizada em 18:17.

5.3 – Resultados Parciais

Como pode ser observado, tanto na primeira fase quanto na segunda, a média de tempo da composição utilizando o FT foi menor do que utilizando o framework OO, mostrando evidências de que a hipótese H_0 mostrada na Tabela 2 talvez pudesse ser refutada se experimentos formais fossem conduzidos.

Se os tempos forem analisados individualmente para cada um dos três passos da composição, tanto na primeira fase quanto na segunda, observa-se que a diferença de tempo relevante ocorre na verdade durante o acoplamento da conexão, e não durante o acoplamento do comportamento da persistência ou dos testes.

Na Tabela 8 é mostrado o tempo médio de cada um dos passos da composição para as duas fases do estudo. Essas médias foram obtidas calculando-se a média das médias já obtidas para cada passo do estudo comparativo. Por exemplo, o tempo médio de

acoplamento do comportamento da persistência da primeira fase é 03:40 com o framework OO (considerando as duas aplicações), como pode ser visto na Tabela 8. Esse valor foi obtido calculando-se a média das médias das duas aplicações com o mesmo framework, não fazendo distinção entre experientes e novatos – 3:42 e 3:37, como pode ser visto na terceira linha na Tabela 5. O mesmo foi feito para os demais valores.

Tabela 8 – Tempo Médio dos Passos de Composição da Primeira e Segunda Fases

Primeira Fase	
Composição com Framework OO	
Média do Tempo de Acoplamento da Persistência com FOO	03:40
Média do Tempo de Acoplamento da Conexão com FOO	**|--- 06:58 ---|**
Média do Tempo de Realização dos Testes com FOO	06:05
Composição com Framework Transversal	
Média do Tempo de Acoplamento da Persistência com FT	03:56
Média do Tempo de Acoplamento da Conexão com FT	**|--- 02:32 ---|**
Média do Tempo de Realização dos Testes com FT	05:20
Segunda Fase	
Composição com Framework OO	
Média do Tempo de Acoplamento da Persistência com FOO	06:07
Média do Tempo de Acoplamento da Conexão com FOO	**|--- 06:32---|**
Média do Tempo de Realização dos Testes com FOO	05:37
Composição com Framework Transversal	
Média do Tempo de Acoplamento da Persistência com FT	03:41
Média do Tempo de Acoplamento da Conexão com FT	**|--- 02:45 ---|**
Média do Tempo de Realização dos Testes com FT	04:52

O passo de acoplamento da persistência possui diferenças muito sutis de tempo entre as versões OO e OA. A justificativa para isso é que a quantidade de linhas de código escritas para realizar esse passo é bastante similar entre as duas versões, como pode ser visto comparando a Figura 1 com a Figura 2. A diferença entre as versões OO e OA é que nessa última, todas as linhas de código estarão em um único aspecto e não espalhadas pelo sistema. Dessa forma, pode-se deduzir que embora determinados interesses transversais tenham um impacto significativo na manutenção e legibilidade do código, influenciam minimamente o tempo da composição.

Com base nesses dados, observa-se que a diferença de tempo significativa somente ocorre no passo de acoplamento da conexão. Na primeira fase do estudo, o tempo médio de realização desse passo usando o FOO foi de 6:58, enquanto que usando o FT foi de 2:32, uma diferença de 63,63%. Na segunda fase a diferença foi de 57,90%.

Observando-se os dados coletados pôde-se concluir que essa diferença de tempo está diretamente relacionada com a quantidade de locais do código que devem ser manipulados, mas principalmente, com a quantidade de linhas de código que são escritas para a realização do acoplamento. Tanto na primeira fase quanto na segunda, houve uma diferença significativa na quantidade de linhas de código escritas quando a conexão é acoplada utilizando-se o FT, fazendo com que o tempo de acoplamento diminua. Comparando-se a Tabela 5 e a 6 pode-se observar que a quantidade de linhas de código escritas (LC) para o acoplamento da aplicação de loja de CDs com o framework OO foi de 48 linhas, enquanto que o acoplamento do FT com essa mesma aplicação utilizou entre 7 e 12 linhas. O mesmo ocorreu com a aplicação Abaco, que utilizou 44 linhas de código para o acoplamento com o framework OO e entre 7 e 12 para o acoplamento com o FT.

Mesmo considerando a escrita de uma linha com orientação a aspectos mais difícil e demorada do que a escrita de uma linha com orientação a objetos, a grande diferença na quantidade de linhas escritas favorece a orientação a aspectos. Contudo, para interesses cuja quantidade de linhas de código escritas para o acoplamento é similar entre as duas versões, a orientação a objetos pode ser beneficiada. Note-se, por exemplo, o passo de

acoplamento da persistência na primeira fase do estudo, que envolveu tanto participantes experientes quanto novatos. A Tabela 8 mostra que a média de tempo usando orientação a objetos (3:40) foi 6,7% mais rápida do que usando orientação a aspectos (3:56). Embora seja uma diferença pequena, está diretamente relacionada com o tamanho das aplicações usadas no estudo. Para aplicações maiores essa diferença tende a aumentar. Note-se também que na segunda fase do estudo, que só envolveu participantes experientes, o tempo de realização do primeiro passo favoreceu a orientação a aspectos. Isso ocorre porque, para esse caso, a dificuldade de se escrever uma linha de código com orientação a aspectos é similar à dificuldade com orientação a objetos.

Os gráficos mostrados na Figura 9 apresentam uma comparação entre o número médio de linhas de código escrito durante a composição com o framework OO e com o FT. Cada coluna representa um passo da etapa de composição: a primeira é o acoplamento da persistência e a segunda o acoplamento da conexão. As colunas brancas representam a quantidade média de linhas de código escritas durante a composição com o framework OO e as cinza com o FT. Sobre cada coluna também é encontrado o tempo médio que os participantes gastaram na realização de cada um dos passos, sem fazer distinção entre experientes ou novatos. Os dois gráficos da parte superior da figura correspondem à primeira fase e os dois da parte inferior correspondem à segunda fase. Note-se que em todos os casos, a quantidade de linhas de código necessárias para se realizar o acoplamento da conexão é maior quando se utiliza o framework OO.

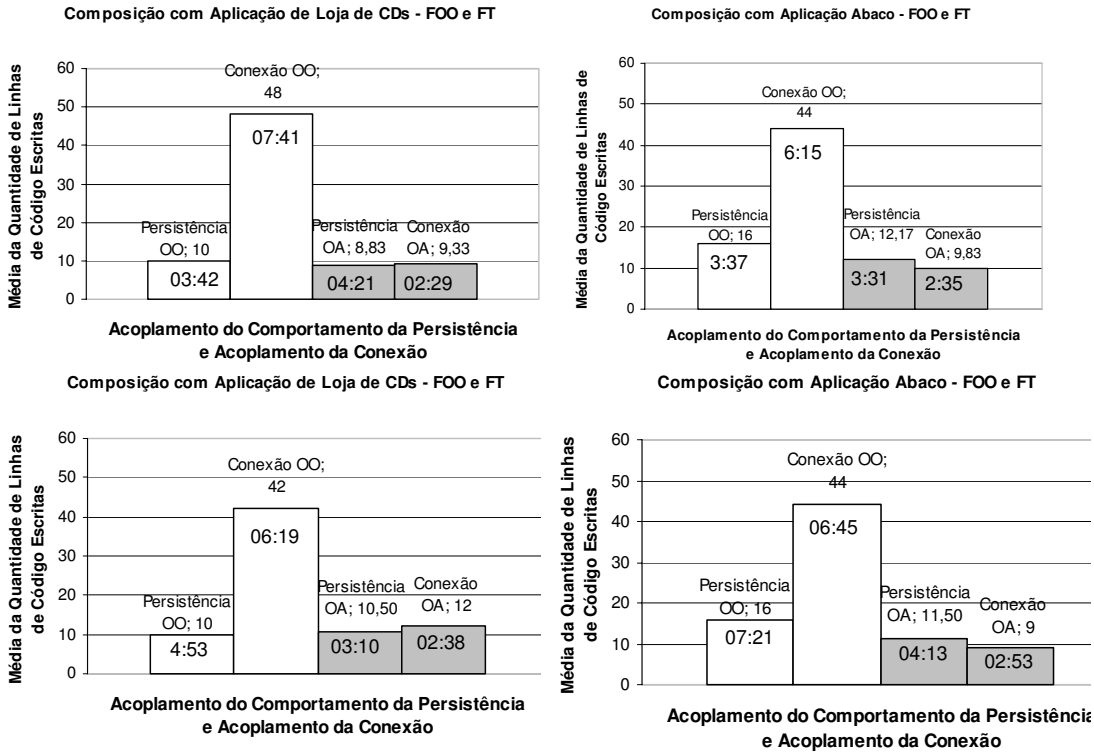

Figura 9 – Comparação do Dado LC

Os resultados da primeira e segunda fase do estudo mostraram uma pequena diferença no tempo de composição a favor do FT, cerca de 4 minutos em média em ambas

as fases, com pode ser observado na Tabela 8. Embora seja uma diferença pequena, o tamanho das aplicações tem influência direta sobre esse número. A composição do framework OO foi em média 30% mais demorada do que com o FT tanto na primeira fase quanto na segunda.

Para muitas organizações essa diferença pode ser insignificante, principalmente para aquelas que trabalham com aplicações de pequeno porte. Além disso, para essas organizações, os recursos investidos no treinamento em uma nova tecnologia pode inviabilizar a adoção de uma nova técnica de programação. Contudo, observa-se que, como essa diferença de tempo está ligada com a quantidade de locais modificados do código, aplicações de grande porte podem se beneficiar com a adoção de um FT, embora novos experimentos devam ser feitos para apresentar dados quantitativos mais sólidos.

Considerando apenas o passo de acoplamento da conexão a diferença percentual é bem maior. Por exemplo, na primeira fase do estudo, a média do tempo de acoplamento da conexão usando o framework OO foi de 6:58, enquanto que com o FT foi de 2:32, uma diferença de 63,64%, um número bastante significativo em se tratando de tempo. Um número relativamente similar é encontrado na segunda fase: 57,91%. Com essa última análise pode-se inferir que determinados interesses transversais que afetam um grande número de locais do código, como ocorre com a conexão, possuem uma diferença significativa no tempo de composição.

6 – Ameaças à Validade do Estudo

Um ponto que pode influenciar os resultados é a forma de implementação dos frameworks. Apesar dos frameworks utilizados no estudo seguirem basicamente a mesma estrutura da maioria dos outros frameworks pesquisados na literatura, pode ser que haja alguma implementação orientada a objetos que diminua o trabalho de acoplamento, e conseqüentemente o tempo total de acoplamento.

O fato de apenas a etapa de composição ter sido submetida à análise não influencia os resultados, pois geralmente a etapa de instanciação de um framework, seja OO ou OA, é feita com mecanismos orientados a objetos. Com base nos resultados obtidos, infere-se que mesmo se a etapa de instanciação de algum FT utilizasse mecanismos orientados a aspectos, o tempo diminuiria, o que não inviabilizaria os resultados. Entretanto, experimentos formais precisam ser conduzidos também nesse sentido.

Outro ponto que pode ter influenciado é a utilização de alunos de pós-graduação como participantes do estudo. Contudo, procurou-se não influenciar os alunos demonstrando expectativas a favor ou contra algum dos frameworks.

7 - Trabalhos Relacionados

Uma série de pesquisas já foi realizada no sentido de mostrar os benefícios de manutenção, legibilidade e reúso que o uso da POA pode trazer para o interesse de persistência [Soares et al., 2002], [Couto et al., 2005], [Rashid and Chitchyan, 2003]. Entretanto, esses trabalhos não apresentam dados quantitativos e experimentos que evidenciem esses benefícios. Além disso, também não comentam se o tempo de composição com o código-base é afetado pelo uso da POA.

Foram encontrados poucos estudos quantitativos envolvendo programação orientada a aspectos em geral. Garcia e outros (2006) fizeram um estudo que procurou averiguar os prós e contras do uso da POA na implementação de padrões de projeto. O trabalho desses autores complementa o trabalho qualitativo anterior de Hanemman e Kiczales (2002) e é baseado nas métricas de coesão, acoplamento e separação de

interesses. Outros estudos experimentais também foram realizados anteriormente, como, por exemplo, o estudo realizado por Kersten e Murphy (1999), que apresenta regras e políticas utilizadas para alcançarem seus objetivos de manutenibilidade, e o trabalho de Zhao e Xu (2004), que apresenta certas métricas que podem ser utilizadas para medir a coesão de um programa orientado a aspectos.

8. Conclusão e Trabalhos Futuros

O contínuo interesse de pesquisadores em frameworks de persistência [Soares et al., 2002], [Couto et al., 2005], [Rashid and Chitchyan, 2003] e a quantidade de frameworks distribuídos gratuitamente e amplamente utilizados [Hibernate, 2006], [OJB, 2006], [Cayenne, 2006] mostram a importância desse tema para a área de engenharia de software. O estudo apresentado por este artigo complementa pesquisas realizadas anteriormente que mostraram que o uso da POA traz benefícios de reúso e manutenção na implementação do interesse de persistência.

Embora o estudo tenha sido realizado com frameworks, os resultados mostram indícios de que o tempo de composição de outros interesses implementados com aspectos, desde que sejam transversais, é menor do que de um interesse implementado com orientação a objetos. Isso se dá pela quantidade de linhas de código que devem ser escritas quando a implementação do interesse se espalha pelos módulos do sistema.

Após a realização do estudo concluiu-se que há diferença no tempo de composição quando se utiliza orientação a aspectos. Em conseqüência da quantidade de linhas de código que deve ser escrita para realizar a composição, a orientação a aspectos é beneficiada pelo mecanismo de quantificação, em que vários pontos de acoplamento são informados em poucas linhas.

Note-se que os dados apresentados por este artigo só são válidos para frameworks caixa branca, em que a composição é realizada com acesso ao código-fonte do framework. Quando há ferramentas que auxiliam no processo de reúso, o tempo de composição passa a ser algo irrelevante, pois será realizado rapidamente independentemente se a implementação é feita com orientação a aspectos ou não [Couto et al., 2005]. Contudo, a maior parte dos frameworks que tratam de um interesse encontrado na literatura não possui apoio automatizado.

A decisão por utilizar um FT desenvolvido pelo próprio grupo de pesquisa se deu pelo acesso livre ao código-fonte, e pelo conhecimento total das funcionalidades disponíveis no framework. Além disso, o acesso ao código-fonte desse FT facilitou o desenvolvimento do framework OO de persistência, já que grande parte do código foi reaproveitada. O desenvolvimento do framework OO com funcionalidades idênticas faz com que o nível de confiabilidade do estudo de caso aumente. Não faria sentido comparar o FT que já estava disponível com o framework OO de persistência *Hibernate*, por exemplo, pois ambos possuem arquiteturas diferentes, com funcionalidades e técnicas de instanciação diferentes. Isso poderia acarretar diferenças favorecendo ou prejudicando o tempo de composição de um dos frameworks.

Como trabalhos futuros, pretende-se replicar o experimento para aplicações maiores, com o objetivo de averiguar se o tempo de composição continua aumentando conforme o tamanho da aplicação. Além disso, pretende-se também realizar o mesmo tipo de experimento com outros frameworks, por exemplo, controle de acesso, autenticação e frameworks que tratam de requisitos funcionais.

Espera-se que o pacote de experimentação, que está disponível em www.icmc.usp.br/~valter, seja utilizado por outros pesquisadores e/ou profissionais em novos experimentos que utilizam aplicações de grande e médio porte.

Referências Bibliográficas

Camargo, V.V., Masiero, P.C. (2005) "Frameworks Orientados a Aspectos". In: anais do 19º Simpósio Brasileiro de Engenharia de Software (SBES'2005), Uberlândia-MG, Brasil, outubro.

Cayenne. (2006). http://www.objectstyle.org/cayenne/ (último acesso: 04 de abril de 2006)

Constantinides, C.A., Bader, A, Fayad, M.F. (2000) "Designing an Aspect-Oriented Framework in an Object-Oriented Environment". ACM Computing Surveys, v.32.

Couto, C.F.M., Valente, M.T.O., Bigonha, R.S. (2005) "Um Arcabouço Orientado por Aspectos para Implementação Automatizada de Persistência". In: anais do 2º. Workshop Brasileiro de Desenvolvimento de Software Orientado a Aspectos (WASP'05), evento satélite do XIX SBES, Uberlândia, MG, Brasil, outubro.

Garcia, A., Santanna, C., Figueiredo E., Kulesza, U., Lucena, C., Staa, A. (2006) Modularizing Design Patterns with Aspects: A Quantitative Study. Transactions On Aspect-Oriented Software Development I, Series: Lecture Notes in Computer Science, Vol. 3880, p. 36-74.

Hanenberg, S., Hirschfeld, R., Unland, R., Kawamura, K. (2004) "Applying Aspect-Oriented Composition to Framework Development – A Case Study". In: proc of 1st International Workshop on Foundations of Unanticipated Software Evolution, Barcelona, Spain, march 28.

Hibernate. (2006) www.hibernate.org (último acesso em 4 de abril de 2006).

Hannemann, J., Kiczales, G. (2002) "Design Pattern Implementation in Java and AspectJ". In: proc. of OOPSLA'02 (November), 161-173.

Kersten, A., Murphy, G. (1999) Atlas: "A Case Study in Building a Web-based learning environment using aspect-oriented programming". In: proc. of OOPSLA'99, November.

Kiczales, G.; Lamping, J.; Mendhekar, A.; Maeda, C.; Lopes, C.; Loingtier, J.; Irving, J. (1997) "Aspect Oriented Programming". In: proceedings of ECOOP. pp. 220-242.

OJB. (2006) http://db.apache.org/ojb/. (último acesso em 4 de abril de 2006).

Pinto, M., Fuentes, L., Fayad, M.E, Troya, J.M. (2002) "Separation of Coordination in a Dynamic Aspect Oriented Framework". In: proc. of the 1st International Conference on Aspect-Oriented Software Development, April.

Rashid, A., Chitchyan, R. (2003) "Persistence as an Aspect". In: proc. of 2nd International C. on Aspect Oriented Software Development(AOSD) Boston–USA, March.

Soares, S., Laureano, E., Borba, P. (2002) "Implementing Distribution and Persistence Aspects with AspectJ". In: proc. of ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications.

Vanhaute, B., Win, B., Decker, B. (2001) "Building Frameworks in AspectJ". In: European Conference on Object-Oriented Programming (ECOOP), Separation of Concerns Workshop. pp. 1-6, June.

Wohlin, C.; Runeson, P.; Höst, M.; Regnell, B.; Wesslén, A. Experimentation in Software Engineering: an Introduction. Kluwer Academic Publishers, 2000.

Zhao, J., Xu, B. (2004) "Measuring Aspect Cohesion". In: Proc of the Conference on Fundamental Approaches to Software Engineering (FASE'2004), LNCS 2984, Springer, Barcelona, Spain, March 29-31, 54-68.