AVALIAÇÃO DE ESFORÇO DE COMPOSIÇÃO DE MODELOS DE SOFTWARE:
UM ESTUDO BASEADO EM NEUROCIÊNCIA

Mateus Manica (mateus.manica@gmail.com)

Resumo: Desenvolvedores usam técnicas de composição de modelos para integrar modelos de software elaborados em paralelo, incluindo técnicas baseada em heurística (IBM RSA, ou Astah, por exemplo) e técnicas baseada em especificação (Epsilon, por exemplo). A tarefa de compor modelos de software não é trivial, visto que os desenvolvedores precisam integrar as partes comuns dos modelos e solucionar as partes conflitantes. Se os conflitos não são solucionados de forma adequada, então abre-se espaço para o surgimento de inconsistências. Desse modo, é necessário investir esforço para detectar e resolver as inconsistências geradas. Caso não sejam resolvidas, tais inconsistências podem, por exemplo degradar arquitetura do software. O problema é que pouco é conhecido sobre o impacto destas técnicas no esforço de composição e na atividade cerebral dos desenvolvedores. Se uma técnica exige muito esforço e leva um desenvolvedor a um quadro de atividade cerebral desfavorável (ou seja, um elevado grau de frustação), então é questionável a adoção desta técnica em projetos reais na indústria. Este estudo, portanto, foca em apresentar um projeto experimental detalhado, bem como produzir evidências empíricas sobre o impacto de tais técnicas no esforço de composição e na atividade cerebral do desenvolvedores. Um experimento controlado foi executado para mensurar o esforço de composição de modelos e para entender as respostas do cérebro dos desenvolvedores ao utilizar tais técnicas. Para isto, foi utilizada uma interface cérebro-máquina.

Palavras-chave: Avaliação de esforço. Técnicas de composição de modelos. Neurociência.

1 INTRODUÇÃO

Os modelos de software estão cada vez mais presentes nos diferentes processos de desenvolvimento de software (RUMBAUGH; JACOBSON; BOOCH, 2004), deste modo expondo os a diferentes formas de uso, análise e tratamento. Em desenvolvimento colaborativo, (WHITEHEAD, 2007), a composição de modelos pode se tornar um desafio devido às recorrentes evoluções a que são submetidos os modelos. Isto ocorre porque os desenvolvedores trabalham paralelamente na elaboração de modelos de software a fim de tornar o processo de desenvolvimento mais ágil, ao passo que reduz o esforço de desenvolvimento. Sendo assim, cada desenvolvedor faz as modificações necessárias nos modelos que usou como referência para o desenvolvimento de uma parte da solução, não levando em conta o uso dos mesmos por outros desenvolvedores. Então, ao final de uma etapa ou

iteração de desenvolvimento é necessário atuar sob os diferentes modelos modificados pelos desenvolvedores envolvidos para compor um modelo final.

A composição de modelos ocorre através de evoluções que agregam novas características, reconciliam modelos criados paralelamente ou adaptam e integram modelos de software. Para efetuar a composição de dois modelos, M_A e M_B, por exemplo, é necessário executar um conjunto de ações para obter o modelo pretendido, M_{AB}, ou seja, o modelo ideal. No entanto, geralmente o modelo composto obtido não é o modelo desejado, M_{CM}, visto que o mesmo possui algum tipo de inconsistência. Essas inconsistências são produzidas a partir de decisões equivocadas e/ou imprecisas por parte dos desenvolvedores durante as atividades de composição de modelos.

Para compor dois (ou mais) modelos é comum o uso de ferramentas, tais como IBM *Rational Software Architect* (RSA), Kompose (Kompose, 2011), MATA, Epsilon e *Eclipse Modeling Framework* (EMF). Essas ferramentas podem ser classificadas de acordo com a técnica empregada para solucionar os conflitos, subdividindo-se em técnica baseada em heurística e técnica baseada em especificação. Ambas visam identificar as diferenças e as semelhanças entre os modelos e tentam também guiar os desenvolvedores na resolução dos conflitos existentes. Contudo, independente da técnica utilizada para compor um modelo o resultado obtido frequentemente pode apresentar inconsistências.

Dado que o modelo gerado pode ter algumas inconsistências, a academia e a indústria têm investido esforços para melhorar as técnicas de composição de modelos (THAKER *et al.*, 2013). Alguns estudos propõem análises semânticas dos conflitos para reduzir a criação de novos conflitos subsequentes, o que aumentaria a eficiência da composição (SHAO *et al.*, 2009). Em estudos mais recentes o foco foi elucidar as razões da baixa eficácia da composição de dois modelos de acordo com a aplicação de cada técnica. Esse estudo verificou que o uso da técnica baseada em heurística exigiu menos esforço que a técnica baseada em especificação para produzir o mesmo modelo desejado (FARIAS *et al.*, 2012). No entanto, não foram observadas diferenças significativas entre os modelos produzidos pelas duas técnicas.

Com base em estudos já realizados no segmento de composição de modelos e nas ferramentas apropriadas para tal finalidade, questiona-se o porquê de tamanha complexidade enfrentada pelos desenvolvedores na solução de conflitos.

Além disso, quais seriam as razões de tomar decisões tão frequentemente incorretas na solução dos conflitos? Sob uma ótica questionadora o presente trabalho buscará entender o impacto destas técnicas no esforço de composição e na qualidade do modelo gerado. Além disso, investiga-se também o impacto destas técnicas na atividade cerebral dos desenvolvedores (em particular nos fatores afetivos dos desenvolvedores de software) e como tais fatores afetam o esforço de composição.

As seguintes seções dividem-se em (2) Referencial Teórico que fundamenta o estudo, (3) Projeto Experimental que apresenta o experimento desenvolvido e seus componentes, (4) Resultados que foram obtidos com o experimento e por fim (5) Conclusões.

2 REFERENCIAL TEÓRICO

A presente seção tem o intuito de posicionar o estudo frente às informações cientificas já disponíveis e garantir melhor compreensão nas seções seguintes através da prefixação de conceitos e conhecimentos. Para isso, a Seção 2.1 introduz o conceito de composição de modelos. A Seção 2.2 descreve as ferramentas de composição de modelos. Por fim, a Seção 2.3 especifica os modelos UML que serão utilizados no estudo.

2.1 Composição de Modelos

Dadas as condições limitadas do ser humano em trabalhar com atividades de grande complexidade (WHITEHEAD, 2007), fazem-se necessárias adaptações nas formas de trabalho. Para tanto, em ambientes de desenvolvimento de software, uma tarefa que tenha maior complexidade é dividida em várias outras tarefas menores e de menor complexidade. Estas tarefas menores e mais bem definidas são direcionadas a diferentes equipes de desenvolvimento que atuam paralelamente. Em outras palavras, as equipes passam a atuar em diferentes partes do modelo de software previamente definido. Como consequência, cada equipe, em sua individualidade, possui pouco conhecimento da solução como um todo, uma vez que se concentraram apenas em algumas partes da mesma. Além disso, as equipes efetuam as modificações que julgam necessárias nos elementos pertinentes as suas tarefas não considerando o uso de alguns destes elementos por outras equipes.

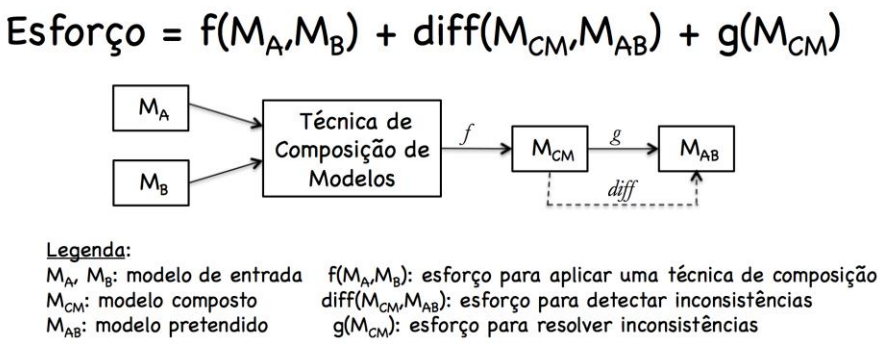
Logo, estas partes da solução que são compartilhadas por diferentes equipes atribuem aos modelos de software diferentes modificações o que torna necessária a composição de modelos entres as versões modificadas por cada equipe. Após a composição, espera-se obter modelos compostos que comtemplem modificações correntes e permitam as futuras. Deste modo habilitando as equipes a atuar sob a solução sem promover, por exemplo, a degradação arquitetural nem propagar inconsistências. Sendo assim, o modelo composto passa a ser o novo modelo de referencia utilizado nas modificações.

No entanto, a tarefa de compor modelos não é tipicamente eficaz. Por exemplo, em uma composição que se dá a partir de dois modelos de entrada, M_A e M_B, espera-se obter um modelo composto M_{AB}, tido como ideal. Contudo, o modelo tipicamente obtido é o M_{CM}, resultado da composição de M_A e M_B, e distinto do ideal M_{AB} devido à presença de inconsistências. Tais inconsistências são oriundas de decisões equivocadas e/ou imprecisas no processo de composição. Deste modo, é necessário mais uma iteração sob os modelos analisados para que se detecte e atue sob as inconsistências com o intuito de obter o modelo M_{AB}.

Existem diversos propósitos para se utilizar a composição de modelos de software. Atualmente três deles recebem foco principal, uma vez que dominam uma ampla quantidade de casos: (1) modificação de modelos; (2) reconciliação de modelos; e (3) análise de sobreposição entre modelos. Todas as situações possuem cenários anteriores muito semelhantes: modelos de software que sofreram modificações colaborativas, ou em paralelo. No entanto, o propósito de cada uma é bastante definido. A modificação de modelos busca principalmente incorporar as modificações de modelos criados em paralelo, sejam elas adição, modificação, remoção ou aprimoramento de elementos. Visto que o propósito é justamente trazer as modificações para o modelo final, os desenvolvedores não avaliam possíveis impactos de suas modificações nem presam pela devida integração das partes. Porém, quando se deseja estabelecer uma reconciliação entre os modelos, é necessário identificar os conflitos e promover o devido ajuste entre os modelos divergentes (CLARKE, 2001). O terceiro caso, análise de sobreposição de modelos, tem por objetivo identificar partes sobrepostas entre os modelos e então definir de qual dos modelos analisados serão mantidas as modificações no modelo final. Os três casos são típicos do ambiente de desenvolvimento e apresentam-se com frequência na composição de modelos.

Neste estudo, os esforços que os desenvolvedores investem para compor modelos se concentram em três atividades: (1) $f(M_A,M_B)$ aplicar as técnicas de composição de modelos; (2) $diff(M_{CM},M_{AB})$ detectar inconsistências no modelo composto; e (3) $g(M_{CM})$ resolver inconsistências encontradas no modelo composto. A Figura 1 a seguir esquematiza a forma como o esforço de composição de modelos é tratado e equacionado neste estudo.

Figura 1 - Equação de esforço de composição de modelos



Fonte: FARIAS *et al.*,(2012)

2.2 Ferramentas de composição de modelos

Para compor modelos são utilizados ferramentas que visam identificar conflitos, orientar o desenvolvedor durante a composição e consolidar os modelos. Estas ferramentas podem ser subdivididas quanto à técnica que empregam na composição dos modelos:

- *Técnica baseada em especificação*: através desta técnica, os desenvolvedores determinam explicitamente como tratar os elementos dos modelos de entrada. Isto inclui a forma como serão avaliados e compostos. Um exemplo de ferramenta que utiliza essa técnica é o Epsilon (EPSILON, 2011) que será detalhado a seguir.
- *Técnica baseada em heurística*: possui um grupo de composições heurísticas predefinidas. Estas composições são responsáveis por avaliar a correspondência entre os elementos e julgar os elementos dos modelos fornecidos para a composição. Uma ferramenta que faz uso desta técnica é o IBM RSA (IBM, 2012) também será detalhada a seguir.

2.1.1 Epsilon

Epsilon é uma completa plataforma para gerenciamento de modelos (KOLOVOS; ROSE; PAIGE, 2011). Com uma família de linguagens e ferramentas para geração de código, transformações entre modelos, refatoração, migração, validação e comparação de modelos é reconhecido como um robusto plug-in para IDE Eclipse. Esse plug-in é desenvolvido com base no *Eclipse Modeling Framework* (EMF), que é amplamente utilizado para o desenvolvimento de novas ferramentas para a plataforma eclipse.

As linguagens disponibilizadas no Epsilon são direcionadas para finalidades específicas, para que os desenvolvedores desempenhem suas atividades de forma bastante precisa para os diferentes propósitos envolvendo os modelos. Ao todo são sete linguagens para diferentes finalidades, no entanto as linguagens de maior interesse do presente estudo são as de comparação (*Epsilon Comparison Language*, ECL) e de fusão ou *merge* (*Epsilon Merge Language*, EML). Estas linguagens baseiam-se em regras claramente definidas para comparar e fundir modelos, que são atividades típicas da composição de modelos (KOLOVOS; ROSE; PAIGE, 2011).

2.2.2 IBM RSA

Como técnica baseada em heurística o IBM *Rational Software Architect* (RSA) é uma ferramenta proprietária e foi considerada no presente fundamentação teórica uma vez que é a ferramenta de modelagem e composição de modelos mais robusta do mercado. Além de ser uma ferramenta robusta e que contempla os artefatos definidos pela UML, foi projetada baseada em estudos empíricos, o que aproxima sua solução dos problemas mais vivenciados nos ambientes de desenvolvimento.

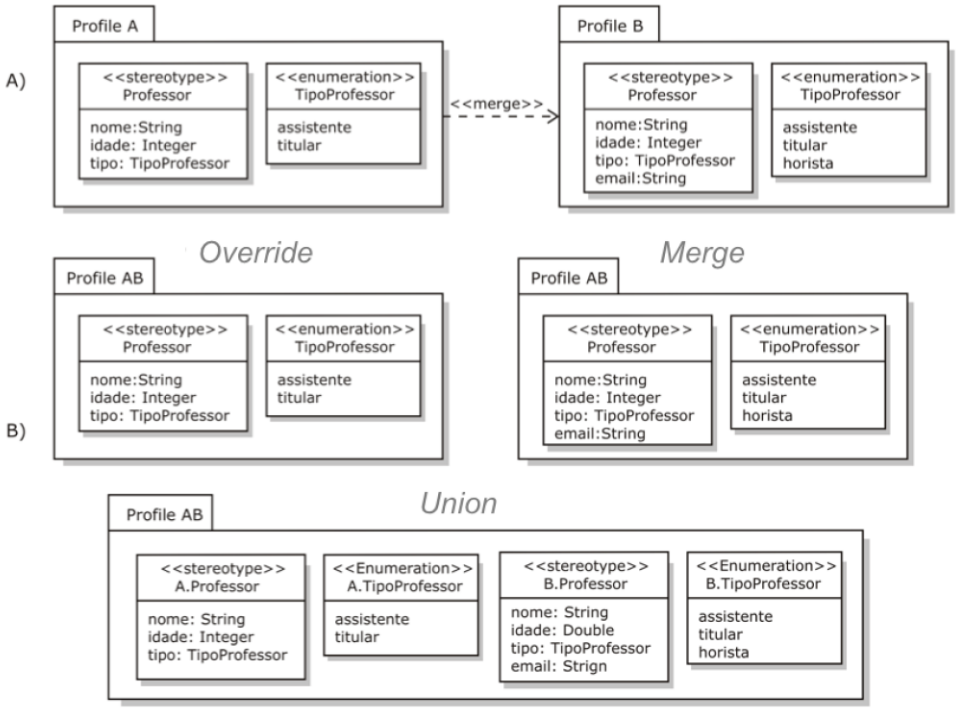
Através de interface amigável, oferece ao usuário a identificação de inconsistências sintáticas, ou seja, o esforço para identificação de conflitos entre os modelos é reduzido. No entanto, nem sempre a técnica empregada pelo IBM RSA é suficiente para produzir o modelo ideal, o que ainda acarreta em tratamento das composições através de interação do usuário para solução dos conflitos. Através de decisões pré-definidas pela própria aplicação, o usuário deve selecionar o que julga como ideal (ou não) para solucionar um dado conflito. O IBM RSA pode ser

considerado como uma ferramenta semiautomática, uma vez que ainda depende das decisões do usuário para solucionar os conflitos.

2.2.3 Algoritmos de composição tradicionais

Além de estudar a ferramenta Epsilon e o IBM RSA, o presente estudo busca analisar também o esforço investido com os algoritmos de composição tradicionais. Estes algoritmos são também exemplos de técnica baseada em heurística, como o IBM RSA. Os algoritmos avaliados são: *override*, *merge* e *union*. Esses algoritmos apresentam-se como um guia para os desenvolvedores com descrições e orientações de como as composições devem ser realizadas. Além disso, são vastamente utilizados em diversos cenários de composição de modelos, como integração de linhas de produtos de software, composição de modelos e modelagem orientada a aspecto. Na Figura 2, a seguir é mostrado o comportamento geral dos métodos de substituição/sobrescrita, merge e união e que serão detalhados em seguida.

Figura 2 - Exemplo de uso do algoritmos de composição



Fonte: OLIVEIRA (2008)

São considerados como modelos de entrada M_A (*Profile A* na Figura 2(a)) e M_B (*Profile B* na Figura 2(a)), nos quais são aplicados cada um dos algoritmos.

Considera-se que os modelos de entrada são correspondentes de acordo com o processo de verificação da equivalência de modelos e, portanto pode-se efetuar o procedimento pertinente a cada algoritmo. Para o método de *override*, Figura 2(b), analisa-se do sentido de M_A para M_B. Onde verifica-se que para todo elemento presente no modelo M_A deve-se substituir seu semelhante (caso haja) no modelo M_B pelo elemento de M_A. Em outras palavras um elemento do modelo M_B (delta de modificações) que possua um correspondente no modelo M_A deve, no modelo composto final, ser representado pelo elemento proveniente de M_A (modelo base). Os elementos que não possuem modificações ou correspondência são igualmente mantidos no modelo final.

Para o algoritmo de *merge*, Figura 2(b), espera-se que para todo elemento correspondente entre os modelos M_A e M_B seja definido no modelo de saída um elemento no qual são consideradas as características do elemento de cada modelo de entrada. Ou seja, o elemento presente no modelo de saída deve considerar o domínio de valores que possa atender tanto as necessidades do elemento de M_A quanto do elemento de M_B, abordagem bastante distinta da empregada pelo algoritmo de substituição. Para tanto, o algoritmo de merge busca identificar o tipo dos elementos de cada modelo de entrada, para então definir um tipo de elemento cujo domínio seja capaz de compreender os valores dos elementos de M_A e M_B. Os elementos que não possuem correspondência entre os modelos analisados são mantidos sem modificações e incluídos no modelo de saída.

Por sua vez, o algoritmo de união, Figura 2(c) considera que todos os elementos correspondentes entre os modelos M_A e M_B devem ser manipulados de modo que se preserve a distinção entre os elementos de M_A e M_B. Logo, no modelo final os elementos devem coexistir, utilizando diferentes identificadores. Os elementos que não obtiveram correspondência entre os modelos ou não foram modificados são inseridos no modelo final

2.3 UML

A UML (*Unified Modeling Language*) é uma linguagem de modelagem para diversos fins da área de engenharia de software. Nesta linguagem são usadas notações gráficas bastante definidas que agregam diferentes significados aos diferentes modelos, uma vez que a notação possui valor semântico. Sendo assim, a

complexidade de uma solução pode ser fracionada através de diversos modelos, o que simplifica a compreensão e define uma linha comum de entendimento em uma equipe de desenvolvimento. Além disso, os modelos são mantidos como referencia de documentação da solução e podem ser reutilizados em projetos posteriores. Atualmente compreende 7 diagramas estruturais e 7 comportamentais, totalizando 14 tipos diferentes de diagramas.

Os diagramas estruturais representam a visualização estática das estruturas dos sistemas, através de objetos, atributos, operações e relacionamentos. Alguns exemplos deste tipo de diagrama podem ser o diagrama de classe e o diagrama de componentes. Já os diagramas comportamentais enfatizam o comportamento do sistema mostrando suas colaborações através dos objetos e mudança de estados internos dos objetos. Exemplos destas representações seriam os diagramas de sequencia e diagramas de estado de máquina respectivamente.

Neste trabalho, o diagrama de classes da UML será utilizado, pois, de acordo com (DOBING; PARSONS, 2006), tal diagrama foi identificado como um dos principais e mais amplamente utilizados na prática. Além destes, os digramas de sequência e de casos de uso também foram identificados com potencial uso no dia-a-dia das equipes de desenvolvimento, no entanto não serão considerados nas análises de composição de modelos frente a maior relevância imprimida pelo diagrama de classes e devido à restrição de tempo para executar um estudo detalhado com tais diagramas. Também são estes três diagramas que mais frequentemente demandam esforços de composição segundo estudo de Norris (NORRIS; LETKEMAN, 2011).

3 PROJETO EXPERIMENTAL

A presente seção descreve as principais características e detalhes do projeto experimental necessários para a execução do experimento controlado. É importante ressaltar que o planejamento experimental aqui descrito é baseado no projeto experimental apresentado em (FARIAS *et al.*, 2011). Sendo assim, a Seção 3.1 descreve os objetivos do estudo, bem como as questões de pesquisa que são investigadas. A Seção 3.2 apresenta as hipóteses que foram testadas durante o experimento controlado e que são confrontadas na seção dos resultados. A Seção 3.3 introduz as variáveis do estudo que foram controladas e analisadas. A Seção 3.4

descreve o contexto que o experimento controlado foi executado. A Seção 3.5 apresenta uma visão geral do procedimento experimental conduzido através do que foi definido nas seções anteriores.

3.1 Objetivos e questões de pesquisa

Este estudo visa avaliar os efeitos de duas técnicas de composição de modelos em três variáveis. A primeira variável é o esforço investido na composição de modelos por parte do desenvolvedor. A segunda é a corretude dos modelos gerados, levando em conta os conflitos oriundos das composições. E a terceira, o índice afetivo e seus componentes, a fim de identificar de que forma o desenvolvedor foi afetado durante a composição com cada técnica. Os efeitos das duas técnicas, baseada em heurística e baseada em especificação, serão explorados através de cenários de evoluções concretos de composição de modelos com o intuito de produzir evidencias empíricas. Devido aos anseios empíricos do estudo, os objetivos são baseados e firmados em acordo com o modelo GQM (*goal, question, metric*) (BASILI; CALDIERA; ROMBACH, 1994), como segue:

> ***Analisar*** *as técnicas de composição de modelos*
> ***com o propósito*** *de investigar seus efeitos*
> ***com respeito*** *ao esforço, corretude e afetividade*
> ***da perspectiva*** *do desenvolvedor*
> ***no contexto*** *de evoluções de modelos de projeto de software.*

Em outras palavras, o estudo investiga o impacto das técnicas de composição de modelos na corretude dos modelos obtidos após a composição de modelos, no esforço empregado pelo desenvolvedor e de que forma o desenvolvedor é impactado afetivamente durante o processo. Deste modo, três Questões de Pesquisa (QP) direcionam este estudo:

- **QP1**: Qual o esforço relativo à composição de dois modelos de entrada utilizando técnica baseada em especificação em relação à técnica baseada em heurística?
- **QP2**: O número de modelos compostos corretamente é maior quando se utiliza técnica baseada em especificação ou técnica baseada em heurística?

- **QP3**: O emprego da técnica baseada em heurística produz um maior impacto na afetividade do desenvolvedor do que a técnica baseada em especificação?

3.2 Formulação das hipóteses

Hipótese 1. Ainda que a técnica baseada em especificação proveja um meio mais sistemático para compor os modelos que a técnica baseada em heurística, não é suficiente para reduzir o esforço empregado pelos desenvolvedores na pratica. Os desenvolvedores investem esforço significativo na especificação das composições e este não é necessariamente convertido em um volume maior de modelos compostos corretamente quando comparado com obtido através do emprego da técnica baseada em heurística. Portanto, forma-se a hipótese de que a técnica baseada em especificação tende a requerer maior esforço para compor modelos de design do que a técnica baseada em heurística. No entanto, não é garantido que a hipótese se confirme. Isto porque as técnicas de especificação auxiliam na combinação o que pode proporcionar uma composição mais rápida dos modelos, ou ainda oferecer vantagens na composição de cenários de evoluções complexos. Neste contexto, a primeira hipótese avalia se a técnica baseada em especificação requer mais esforço por parte dos desenvolvedores do que a técnica baseada em heurística. De maneira formal, tem-se a seguinte hipótese:

> **Hipótese Nula 1, $H_{1\text{-}0}$**: A técnica de composição baseada em especificação requer um menor esforço (ou igual a) por parte do desenvolvedor que a técnica de composição baseada em heurística para produzir o modelo M_{AB} a partir de M_A e M_B.
>
> $H_{1\text{-}0}$: $Esfor(M_A,M_B)_{Especificação} \leq Esfor(M_A,M_B)_{Heurística}$
>
> **Hipótese Alternativa 1, $H_{1\text{-}1}$**: A técnica de composição baseada em especificação requer mais esforço por parte do desenvolvedor que a técnica de composição baseada em heurística requer para produzir o modelo M_{AB} a partir de M_A e M_B.
>
> $H_{1\text{-}1}$: $Esfor(M_A,M_B)_{Especificação} > Esfor(M_A,M_B)_{Heurística}$

Dada a verificação desta hipótese poderá ser avaliado se os tipo das técnicas de composição tem papel fundamental na redução do esforço empregado na produção do modelo desejado, M_{AB}. Além de gerar evidências empíricas sobre como

estas técnicas adaptam as modificações de M_B para M_A, por exemplo. Com o conhecimento de qual técnica consome menos esforço, os desenvolvedores podem escolher adequadamente a mais apropriada para um cenário de evolução em particular, de forma que se evite considerar apenas pareceres de desenvolvedores mais experientes sobre o tema. Também foram produzidas outras hipóteses para detalhar as potenciais descobertas, estas podem ser vistas na Tabela 1.

Hipótese 2. Como anteriormente mencionado, o modelo produzido pela composição geralmente apresenta inconsistências. Dependendo da quantidade de inconsistências presentes no modelo M_{CM}, pode-se fazer necessário um esforço extra e elevado para produzir M_{AB} a partir do M_{CM}. Este esforço extra pode ser justificado, por exemplo, pelas etapas de detecção e resolução das inconsistências. Caso se confirme, a demanda de um elevado esforço a eficiência e eficácia da técnica em questão pode ser questionada. O principal argumento para se utilizar a técnica baseada em especificação é que ela permite produzir um grande número de modelos corretamente compostos em acordo com o que foi definido pelo desenvolvedor na etapa de especificação. Isto é, a forma de comparar e compor os modelos são previamente definidas pelo desenvolvedor através de regras dedicadas para estes fins. Tal característica não está presente na técnica baseada em heurística. Em contrapartida, a técnica baseada em heurística supõe as possíveis equivalências entre os modelos de entrada e então os integra, forma mais simplificada que a proposta pela técnica baseada em especificação. Apesar disso, a heurística pode compor os modelos de forma equivocada. Sobretudo não é claro se a abordagem, baseada em especificação, que transfere ao desenvolvedor a responsabilidade de definir as regras de comparação e composição é a que demanda menor esforço. Isto porque uma vez que alguma das regras não esteja precisamente definida os modelos gerados podem ser incorretamente compostos e relacionados. No entanto, ainda que algumas dificuldades sejam enfrentadas pelos desenvolvedores para definir precisamente cada regra, conjectura-se que são capazes de criar regras de integração e comparação que representem os diferentes cenários de evolução. Sendo assim, supõe-se que, através da técnica baseada em especificação, os desenvolvedores podem produzir mais modelos corretos do que se utilizassem a técnica baseada em heurística. Portanto, a segunda hipótese busca avaliar se a técnica de composição baseada em especificação pode levar o desenvolvedor a obter um maior numero de modelos corretamente (Cor) compostos

quando comparada ao uso da técnica baseada em heurística. Deste modo, as seguintes hipóteses nula e alternativa são produzidas:

> **Hipótese Nula 2, $H_{2\text{-}0}$**: A técnica de composição baseada em especificação produz um número menor ou igual de modelos corretamente compostos que a técnica de composição baseada em heurística.
>
> $H_{2\text{-}0}$: $Cor(M_{CM})_{Especificação} \leq Cor(M_{CM})_{Heurística}$
>
> **Hipótese Alternativa 2, $H_{2\text{-}1}$**:A técnica baseada em especificação produz um número maior de modelos corretamente compostos que a técnica de composição baseada em heurística.
>
> $H_{2\text{-}1}$: $Cor(M_{CM})_{Especificação} > Cor(M_{CM})_{Heurística}$

A definição de um modelo como correto é determinada pela presença ou não de inconsistências no modelo composto. Justamente a quantidade destas inconsistências que é o alvo da investigação a respeito das duas técnicas avaliadas. Isto é, se a técnica baseada em especificação implica (ou não) em menor taxa inconsistências do que a técnica baseada em heurística. Com a verificação desta segunda hipótese é possível identificar qual técnica produz uma menor taxa de inconsistência nos modelos compostos. Esta informação subsidia uma melhor escolha de qual técnica deve ser escolhida pelo desenvolvedor uma vez que se sabe qual técnica é mais propensa a produção e propagação das inconsistências. As hipóteses 1 e 2, bem como suas hipóteses auxiliares, são representadas na Tabela 1.

Após refletir e reavaliar as investigações provenientes das duas hipóteses anteriores, que questionam o esforço empregado na composição (hipótese 1) e o quão correto (hipótese 2) estão os modelos gerados por cada técnica de composição, identificou-se a necessidade de adicionar um foco extra a pesquisa. Isto porque ambas as hipóteses avaliam diretamente o impacto das técnicas nos modelos produzidos, não levando em conta outros agentes do cenário de composição. Então, com o intuito de amplificar os resultados do presente estudo, expor um novo meio e forma de pesquisa para a área de conhecimento em questão o fator humano é adicionado ao escopo de exploração. É bom lembrar que o fator humano, ou um conjunto de fatores humanos impactantes no contexto da composição de modelos é representando pelo desenvolvedor. Também é importante

salientar que o desenvolvedor é um agente potencialmente decisivo no processo de composição e que enquanto atua sob os modelos o próprio passa por transformações psíquicas produzidas pelo ambiente externo a si mesmo. Isto significa que durante o processo de composição ocorrem mudanças frequentes na percepção do desenvolvedor que são tão inerentes ao individuo que não as percebe. Logo, é sob esta ótica que se buscará investigar o impacto de cada técnica de composição de modelos no desenvolvedor, através das alterações dos indicadores de afetividade.

Hipótese 3. Durante a composição de modelos o desenvolvedor pode ser submetido a diferentes experiências de uso, o que produz percepções instantâneas em relação à atividade, logo o influenciando de forma afetiva. Segundo Maluf (2012, p.19) no campo da psicologia o termo afetividade define a suscetibilidade do ser humano perante experiências provocadas por alterações no mundo exterior ou em si próprio. Ainda de acordo com Maluf (2012, p.19) é “Um conjunto de fenômenos psíquicos que se manifestam sob a forma de emoções, sentimentos e paixões acompanhados sempre da impressão dor ou prazer, satisfação ou insatisfação, agrado ou desagrado, de alegria ou tristeza”. Por meio destas definições, questiona-se de que forma, de fato, as técnicas de composição impactam a afetividade dos desenvolvedores. Se o desenvolvedor se sente insatisfeito ao utilizar uma técnica de composição, então é questionável se a mesma lhe ajudará a produzir o modelo composto esperado, ao passo que reduz o esforço de composição.

Fatores como a usabilidade, controle sob a atividade, retorno (*feedback*) sob o que é feito, volume das inconsistências produzidas em um modelo, forma de uso das técnicas, avaliação dos resultados entre outros são candidatos a influenciar afetivamente o desenvolvedor. Por exemplo, na técnica baseada em especificação se poderia esperar uma maior concentração por parte do desenvolvedor já que o obriga refletir mais intensamente sobre os modelos envolvidos para fazer as corretas definições nas regras. Também pode-se esperar de alguns desenvolvedores uma maior excitação devido ao apreço genuíno a uma de suas funções primordiais, programar. No entanto, pode haver frustração já que não possui um *feedback* instantâneo como ocorre na técnica baseada em heurística. Ou se após um longo período investido na especificação o volume de inconsistências apresentado for desmotivador. Em contrapartida, a técnica baseada em heurística pode oferecer maior engajamento devido à ação direta sob os modelos. Em processos

semiautomáticos pode ter menores índices de frustração já que o esforço não foi investido no mesmo nível de detalhe que a técnica baseada em especificação. Também caso a quantidade de inconsistências seja pequena ou nula pode aumentar o indicador de excitação por desempenhar uma atividade que não é típica do seu dia-a-dia, mas apresentou resultados satisfatórios para o mesmo. Com ciência dos diversos cenários possíveis, conjectura-se, sobretudo que a técnica baseada em especificação pode apresentar menor impacto afetivo sobre o desenvolvedor, uma vez que a tarefa lhe parece corriqueira pelo uso de linguagens de programação e então pode confiar nas próprias habilidades para resolver os conflitos. Atribuindo um posicionamento formal, tratamos então da seguinte hipótese:

> **Hipótese Nula 3, $H_{3\text{-}0}$**: O uso da técnica de composição baseada em especificação tem menor ou igual impacto afetivo nos desenvolvedores que o uso da técnica baseada em heurística.
>
> $H_{3\text{-}0}$: $Afet(M_{CM})_{Especificação} \leq Afet(M_{CM})_{Heurística}$
>
> **Hipótese Alternativa 3, $H_{3\text{-}1}$**: O uso da técnica de composição baseada em especificação tem maior impacto afetivo nos desenvolvedores que o uso da técnica baseada em heurística.
>
> $H_{3\text{-}1}$: $Afet(M_{CM})_{Especificação} > Afet(M_{CM})_{Heurística}$

Além da hipótese 3 definida, são avaliadas outras quatro hipóteses auxiliares baseada nos diferentes indicadores afetivos que, juntos, compõe o índice de afetividade. Sobretudo, é importante ressaltar que os indicadores de frustração, engajamento, meditação, excitação e frustração em longo prazo podem variar diferentemente ao longo da execução de composição de modelos com ambas as técnicas. Por meio da investigação da afetividade e seus indicadores sobre o desenvolvedor, deseja-se esclarecer sob a ótica das técnicas empregadas qual impacta mais afetivamente o usuário. Deste modo, abre-se espaço para novas pesquisas e explorações no âmbito da composição de modelos de software, além de oportunizar a compreensão de como o desenvolvedor percebe a experiência de compor os modelos com cada técnica. Na Tabela 1 podem ser verificadas as hipóteses e suas hipóteses auxiliares como um resumo.

Tabela 1 - Hipóteses

Hipótese Nula	Hipótese Alternativa

$H1_{1\text{-}0}$: $Esfor(M_A,M_B)_E \leq Esfor(M_A,M_B)_H$	$H1_{1\text{-}1}$: $Esfor(M_A,M_B)_E > Esfor(M_A,M_B)_H$
$H1_{2\text{-}0}$: $f(M_A,M_B)_E \leq f(M_A,M_B)_H$	$H1_{2\text{-}1}$: $f(M_A,M_B)_E > f(M_A,M_B)_H$
$H1_{3\text{-}0}$: $diff(M_{CM},M_{AB})_E \leq diff(M_{CM},M_{AB})_H$	$H1_{3\text{-}1}$: $diff(M_{CM},M_{AB})_E > diff(M_{CM},M_{AB})_H$
$H1_{4\text{-}0}$: $g(M_{CM})_E \leq g(M_{CM})_H$	$H1_{4\text{-}1}$: $g(M_{CM})_E > g(M_{CM})_H$
$H2_{1\text{-}0}$: $Cor(M_{CM})_E \leq Cor(M_{CM})_H$	$H2_{1\text{-}1}$: $Cor(M_{CM})_E > Cor(M_{CM})_H$
$H2_{2\text{-}0}$: $Tax(M_{CM})_E \geq tax(M_{CM})_H$	$H2_{2\text{-}1}$: $tax(M_{CM})_E < tax(M_{CM})_H$
$H3_{1\text{-}0}$: $Afet(M_{CM})_E \leq Afet(M_{CM})_H$	$H3_{1\text{-}1}$: $Afet(M_{CM})_E > Afet(M_{CM})_H$
$H3_{2\text{-}0}$: $Engaj(M_{CM})_E \leq Engaj(M_{CM})_H$	$H3_{2\text{-}1}$: $Engaj(M_{CM})_E > Engaj(M_{CM})_H$
$H3_{3\text{-}0}$: $Medit(M_{CM})_E \geq Medit(M_{CM})_H$	$H3_{3\text{-}1}$: $Medit(M_{CM})_E < Medit(M_{CM})_H$
$H3_{4\text{-}0}$: $Excit(M_{CM})_E \geq Excit(M_{CM})_H$	$H3_{4\text{-}1}$: $Excit(M_{CM})_E < Excit(M_{CM})_H$
$H3_{5\text{-}0}$: $Frust(M_{CM})_E \leq Frust(M_{CM})_H$	$H3_{5\text{-}1}$: $Frust(M_{CM})_E > Frust(M_{CM})_H$

Onde:

Esfor: esforço para compor modelos de entrada (QP1), E: técnica baseada em especificação, H: técnica baseada em heurística, f: esforço para aplicar a técnica de composição (QP1), diff: esforço para detectar inconsistências (QP1), g:esforço para resolver as inconsistências (QP1), Cor: corretude da composição (QP2), Tax: taxa de inconsistência do modelo composto (QP2), Afet: afetividade (QP3), Engaj: engajamento (QP3), Medit: meditação (QP3), Excit: excitação(QP3), Frust: frustração (QP3)

Fonte: elaborado pelo autor

3.3 Variáveis de estudo

As variáveis dependentes da hipótese 1 são as responsáveis pela definição do esforço. Então de acordo com o Tabela 1, estas variáveis são *Esfor(M_A,M_B)* que representa o esforço completo ou geral para composição de dois modelos, *f(M_A,M_B)*, o esforço para aplicação da técnica, *diff(M_{CM},M_{AB})*, que é o esforço necessário para identificar as inconsistências e *g(M_{CM})*, esforço necessário para resolver as inconsistências. Todas essas variáveis são medidas em minutos. A escolha destas variáveis se dá pelo fato de serem as etapas pelas quais os modelos passam até estarem compostos, são também as principais atividades executadas pelos desenvolvedores quando em processo de composição de modelos (MENS, 2002). A avalição destas variáveis permite observar o impacto individual de cada etapa, o que permite comparar as duas técnicas quanto a uma etapa específica que tenha se

mostrado crítica. Além disso, é possível verificar qual etapa compromete mais esforço no processo de composição.

Na hipótese 2, as variáveis dependentes definem o quão correto são os modelos gerados e a taxa de inconsistências observada para cada técnica. Na primeira hipótese em respeito da corretude ($H2_1$), espera-se avaliar entre as técnicas qual produziu o modelo resultante mais correto, para casos onde não haja inconsistências define-se que M_{CM} é igual ao modelo idealmente esperado M_{AB}, o qual não demanda esforços extras para finalização da composição. Para definir o quão correto estão os modelos, são feitas comparações entre os modelos gerados e modelos reais, retirados do mesmo sistema de onde foram extraídos os modelos que deram origem a composição. Modelos que apresentarem qualquer tipo de inconsistência seja ela semântica ou sintática é considerado incorreto. Na segunda hipótese que atua sob a qualidade dos modelos gerados ($H2_2$), investiga-se a taxa de inconsistência dos modelos que foram considerados errados. A taxa de inconsistência é definida dividindo-se a quantidade de inconsistências pela quantidade de elementos presente no modelo composto. Dessa forma é possível identificar a densidade de inconsistências de um dado modelo composto. Através da comparação das taxas de inconsistências obtidas após o uso de ambas as técnicas é possível verificar qual técnica que é capaz de produzir os modelos mais próximos do ideal.

As variáveis dependentes da terceira hipótese (H3), informam de que forma o desenvolvedor foi impactado afetivamente pelo uso de cada técnica. A afetividade, *Afet(M_{CM})*, é o resultado da soma dos quatro outros indicadores afetivos mensurados, o que o torna o resultado do impacto total sobre o desenvolvedor. O engajamento, *Engaj(M_{CM})* é o resultado da experiência no estado de alerta, da atenção, direcionada e consciente para a tarefa, quando seus níveis são apresentados negativamente podem significar uma branda vigilância ou até mesmo tédio. A meditação, *Medit(M_{CM})*, informa o quão calmo ou relaxado o usuário esteve durante a prática. A excitação, *Excit(M_{CM})*, determina no trabalho uma condição de alerta sensorial e prontidão de resposta. A frustração, *Frust(M_{CM})*, reflete a reação aos feedbacks provenientes da prática, pode expressar até mesmo stress. Estas informações são obtidas através de um eletroencefalograma (Emotiv EPOC EEG) associado a uma API especifica que trata as informações obtidas das ondas cerebrais em tempo real as transcrevendo para uma aplicação que pode armazená-

lo. Estas informações são medidas de potencial elétrico, volts (V), na ordem dos microvolts (μV) e podem ser gravadas, permitindo a analise.

As variáveis independentes das hipóteses 1, 2 e 3 são as técnicas de composição de modelos: baseada em especificação e em heurística. Controla-se o emprego das técnicas nos cenários propostos para que se observe o impacto nas variáveis dependentes.

3.4 Contexto e Seleção de Participantes

Os participantes do estudo utilizaram duas ferramentas (isto é, Astah e Epsilon) que empregam as duas técnicas de composição de modelos. As ferramentas serão utilizadas para compor os modelos de seis cenários de evolução com diferentes modelos com os quais nenhum dos participantes deve estar familiarizado. A Tabela 2 mostra os cenários de evolução, com sua respectiva descrição dos casos que os desenvolvedores devem utilizar como base para efetuar as composições entre os modelos. É importante destacar que os indivíduos avaliados não estiveram envolvidos previamente no desenvolvimento de nenhum dos casos utilizados no estudo. E também que os cenários utilizados no estudo são fragmentos de modelos industriais captados de diferentes domínios de aplicação, como simulação de extração de petróleo e sistemas de gestão de cadeia de suprimentos.

O experimento foi realizado com 3 participantes tiveram como requisito mínimo: formação em ensino superior ou estar cursando bacharel na área de informática. Portanto espera-se conhecimento em programação e modelagem de software por parte dos participantes (FARIAS *et al.*, 2012). Antes de iniciar as execuções dos experimentos os participantes são todos introduzidos e treinados para compreender e utilizar as técnicas de composição de modelos.

Tabela 2 - Tarefas dos Cenários de Evolução

Tarefa	Modelos	Modificações esperadas nos modelos base
1	Extração de óleo	Adicionar uma classe, um método e um relacionamento. Modificar uma classe de concreta para abstrata
2	Sistema veicular	Remover dois métodos e modificar uma direção de relacionamento

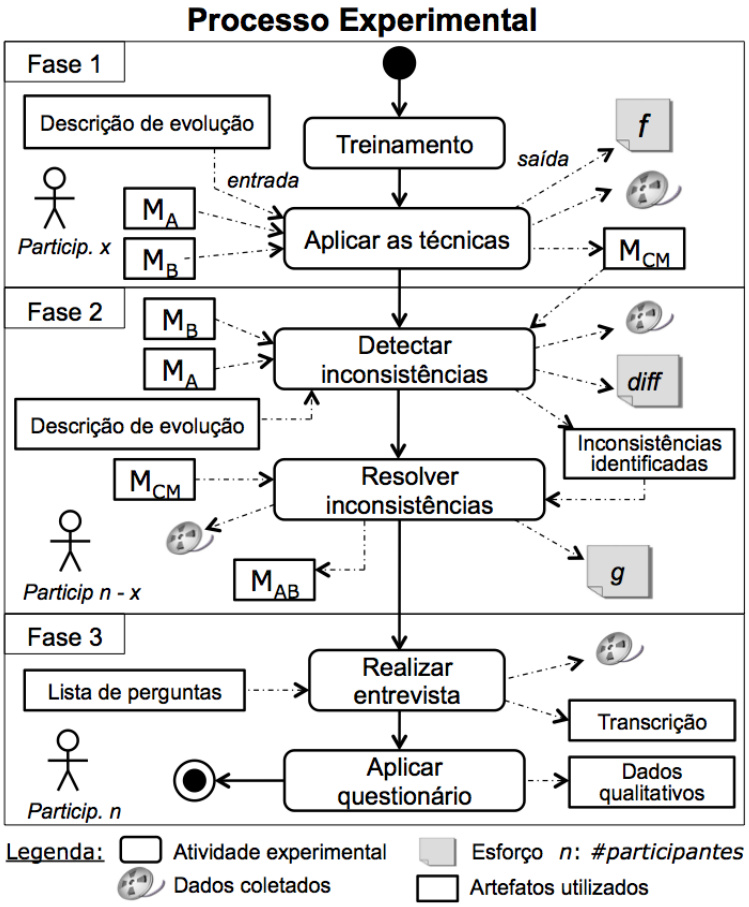
3	ATM	Adicionar duas classes e refinar duas classes a partir de uma. Remover esta ultima classe.
4	Gestão de cadeia de suprimentos	Adicionar duas classes e um relacionamento
5	Gestão financeira	Remover uma classe e adicionar dois métodos para uma classe em particular. Refinar duas classes a partir de uma. Remover esta ultima. Remover um relacionamento.
6	Simulação de extração de petróleo	Modificar a direção de 5 relacionamentos. Modificar o nome de dois métodos

Fonte: elaborado pelo autor

3.5 Processo Experimental

O processo experimental deste trabalho é sumarizado na Figura 3. Esse processo pode ser caracterizado como um bloco único ao acaso, com três tratamentos, o uso de três ferramentas. O estudo possui um conjunto de atividades organizadas em três fases (ver Figura 3). Os participantes serão atribuídos aleatoriamente e distribuídos igualmente aos tratamentos, seguindo um *within-subjects design*, em que todos os participantes do estudo executarão todas as atividades referente aos três tratamentos (WOHLIN, 2000). Em cada tratamento, os indivíduos usarão uma técnica de composição de modelos para realizar duas tarefas experimentais (Tabela 1), totalizando seis tarefas executadas por participante. Portanto, o delineamento experimental foi, por definição, um design balanceado. A Figura 3 mostra através de um processo experimental como as três fases foram organizadas. Os participantes realizaram individualmente todas as atividades para evitar qualquer ameaça ao processo experimental. A seguir o detalhe de cada atividade do processo. Todos os participantes serão treinados para garantir que terão familiaridade com cada técnica de composição de modelos. Os participantes serão incentivados a aplicar as técnicas para compor, M_A e M_B, de acordo com as descrições das modificações (Tabela 1) onde é definido como os elementos dos modelos base, M_A, foram alterados. É importante destacar que o modelo delta, M_B, também já está previamente alterado. A medida de esforço da aplicação das técnicas é obtida em minutos durante a execução da atividade.

Figura 3 - Processo Experimental



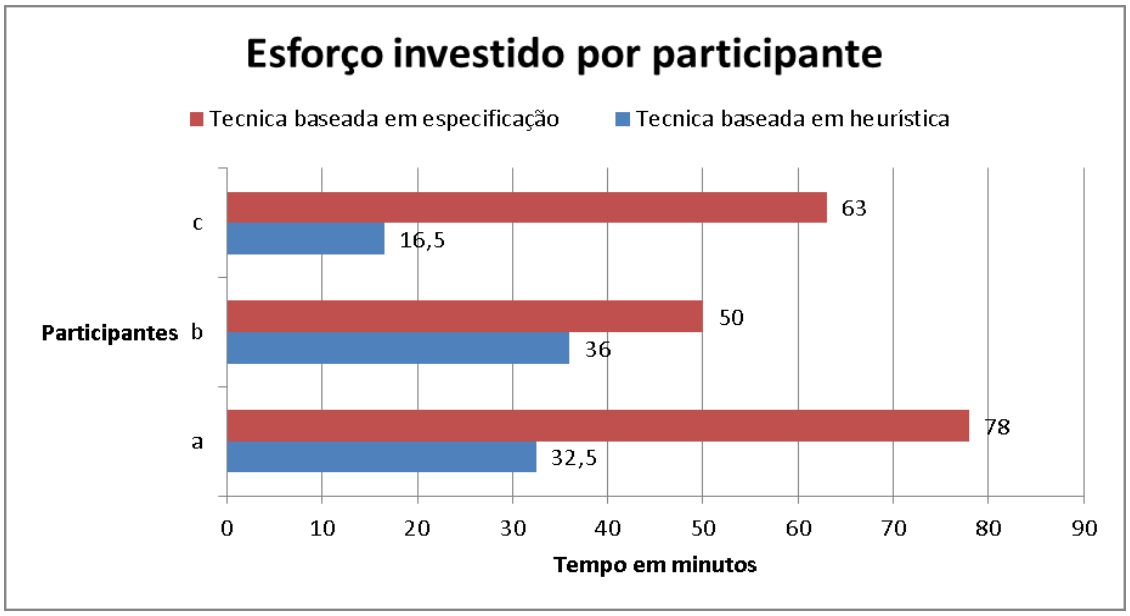
Fonte: FARIAS *et al*. (2012)

4 RESULTADOS

Após apresentar o procedimento experimental utilizado neste estudo, esta Seção foca em apresentar os resultados obtidos considerando os questionamentos anteriormente mencionados (Seção 3.1) e as hipóteses previamente levantadas (Seção 3.2).

Avaliação do esforço de composição (H1). A primeira hipótese (H1) avalia o esforço investido com a técnica baseada em especificação em comparação com a técnica baseada em heurística para realizar a composição de modelos. Além disso, questiona-se até que ponto o uso de técnicas de composição de modelos baseada em especificação potencializa de fato a produção dos modelos integrados sem inconsistências. A Figura 4 apresenta os dados coletados considerando o esforço de composição. É importante destacar que o esforço foi quantificado em minutos e três voluntários participaram do estudo.

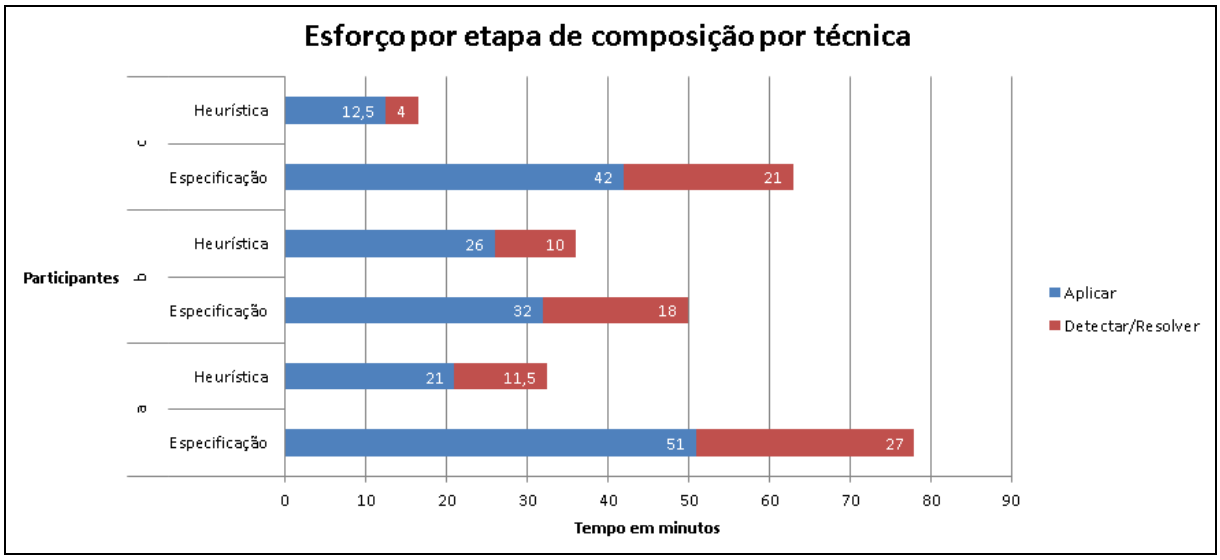
Figura 4 – Esforço investido por participante



Fonte: elaborado pelo autor

Conforme ilustrado na Figura 4, os dados sugerem que os participantes do estudo investiram um maior esforço quando utilizaram a técnica de composição baseada em especificação. O participante “a”, por exemplo, investiu 78 minutos e 32 minutos para integrar modelos usando a técnica baseada em especificação e a técnica baseada em heurística, respectivamente. Ao utilizar a técnica baseada em heurística, o participante investiu apenas 41% do tempo exigido para integrar os modelos utilizando a técnica baseada em especificação. Ou seja, ao utilizar o Epsilon o participante investiu mais que dobro de tempo para produzir o modelo integrado. Observando os dados coletados dos outros participantes se verifica esta mesma tendência de crescimento utilizando a técnica baseada em especificação. Ainda assim, considera-se relevante análise do tempo investido em cada uma das etapas do processo de composição de modelos de software. Na Figura 5 é possível distinguir o tempo investido em cada etapa do processo, isto é, ao aplicar a técnica de composição, detectar e resolver inconsistências.

Figura 5 – Esforço por etapa de composição por técnica



Fonte: elaborado pelo autor

Cabe ressaltar que não foi observada distinção entre a atividade de detectar as inconsistências e resolve-las, já que os participantes solucionavam as inconsistências assim que as detectavam. Além disso, ambas as técnicas apresentavam os modelos compostos de forma bastante expressiva, o que facilita a detecção das inconsistências quando confrontadas com o modelo ideal. Logo o tempo investido na detecção é significativamente inferior ao tempo de resolução das mesmas. No entanto, para fins de análise, documentação e conformidade com as hipóteses auxiliares, manteve-se o registro de detecção junto ao de resolução.

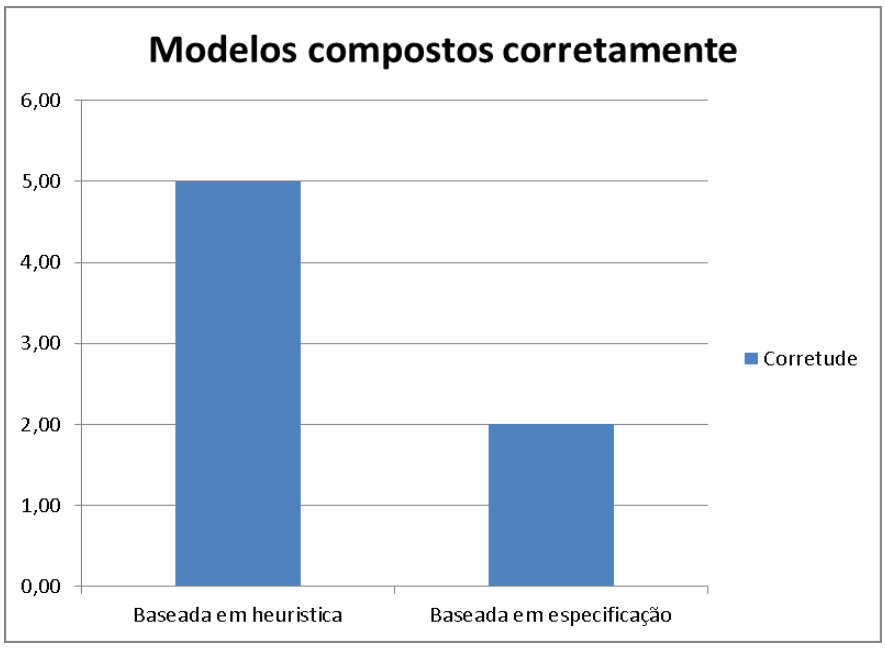
Como resultado preliminar da hipótese H1, observou-se que para todos os participantes houve maior esforço na aplicação da técnica baseada em especificação, o que sugere a rejeição da hipótese nula ($H1_{1\text{-}0}$) e confirmação da hipótese alternativa ($H1_{1\text{-}1}$). Também é possível constatar que a técnica baseada em especificação exigiu maior esforço para detectar e resolver as inconsistências, também confirmando as hipóteses alternativas auxiliares de H1.

Avaliação do corretude das composições (H2).Conforme observado nos resultados anteriores, é frequente o surgimento de inconsistências oriundas do processo de composição de modelos. Isto é, após compor os modelos M_A e M_B geralmente se obtém o modelo M_{CM}, distinto do ideal M_{AB}, devido à presença de inconsistências. Logo, o esforço extra do processo de detecção e resolução das inconsistências pode ser justificado, pois uma vez assumidas as inconsistências,

estas podem se propagar pela arquitetura do software propagando novas inconsistências decorrentes do processo de composição inicial.

Ao utilizar um técnica baseada em especificação, espera-se que o número de modelos produzidos corretamente, M_{CM} = M_{AB}, seja potencializado, dada a flexibilidade que o desenvolver tem de especificar/adequar a forma que a integração tem que realizada frente as requisições de mudanças encontradas no modelo delta, M_B. Porém, não é clara a eficiência nem eficácia de uma técnica em comparação a outra. Para quantificar o impacto das inconsistências, a segunda hipótese (H2) avalia a taxa de inconsistências e a corretude dos modelos gerados por cada uma das técnicas. Na Figura 6, pode-se observar a quantidade de modelos compostos corretamente com cada técnica.

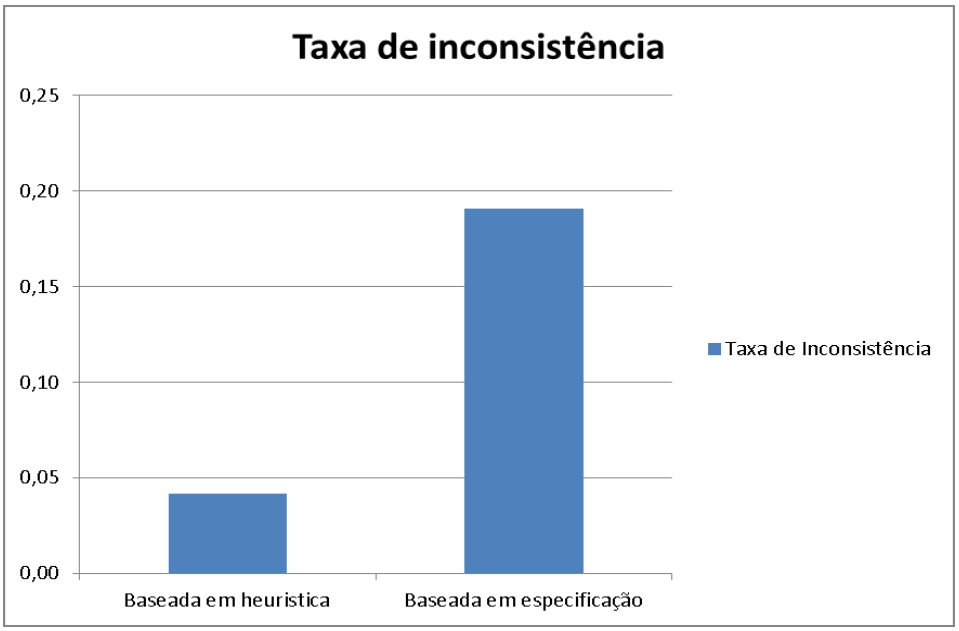
Figura 6 – Modelos compostos corretamente



Fonte: Elaborado pelo autor

De acordo com os dados levantados no procedimento experimental, a técnica baseada em especificação produziu um menor número de modelos integrados de forma correta em comparação ao numero de modelos produzido corretamente com a técnica baseada em heurística, considerando o total das evoluções executadas pelos participantes. Para complementar os fatores qualitativos, a Figura 7 apresenta os dados relacionados à taxa de inconsistência.

Figura 7 – Taxa de inconsistência



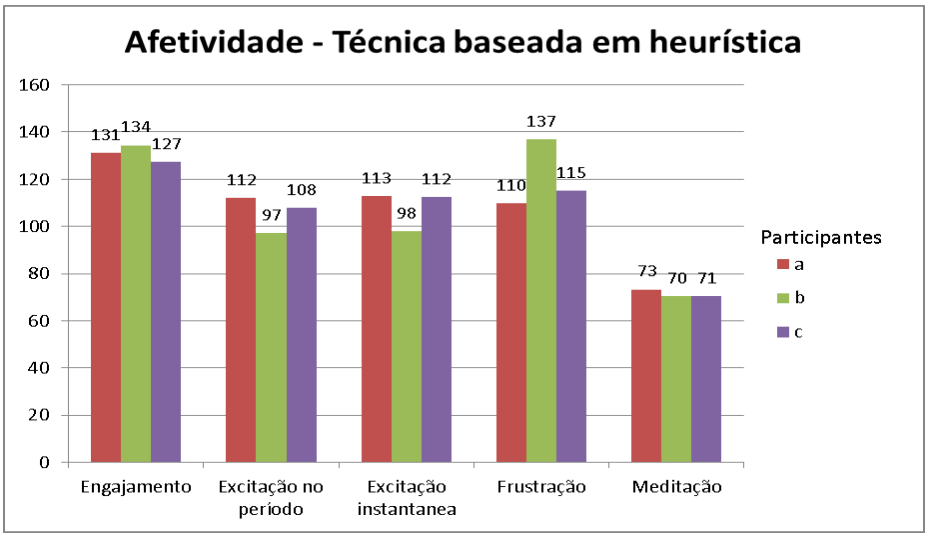
Fonte: Elaborada pelo autor

Com as informações apresentadas na Figura 7, torna-se ainda mais expressivo os resultados da técnica baseada em heurística, uma vez que além produzir maior quantidade de modelos compostos, produziu menor quantidade de inconsistência por elementos. Deste modo, confirma-se a hipótese nula H2, complementada ainda pela confirmação da hipótese nula auxiliar de H2.

Avaliação dos efeitos de composição na afetividade (H3). Para as diversas atividades presentes no processo de desenvolvimento de software os desenvolvedores levam consigo seus sentimentos (ou afetividade), isto é, sua situação emocional individual em um dado momento. Portanto, não seria diferente na etapa de composição de modelos de software, uma vez que este fator é inerente ao ser humano e não exclusivo de uma dada atividade, seja ela qual for. Sendo assim, a terceira hipótese (H3) investiga o impacto das técnicas de composição na afetividade dos desenvolvedores.

Com a totalização dos indicadores afetivos de cada um dos participantes pode-se verificar quais técnicas causaram um maior impacto na afetividade dos desenvolvedores (Figura 12). Além disso, é possível observar os indicadores que compõem a variável afetiva de cada participante, de acordo com cada técnica de composição de modelos de software (Figura 8 e Figura 10).

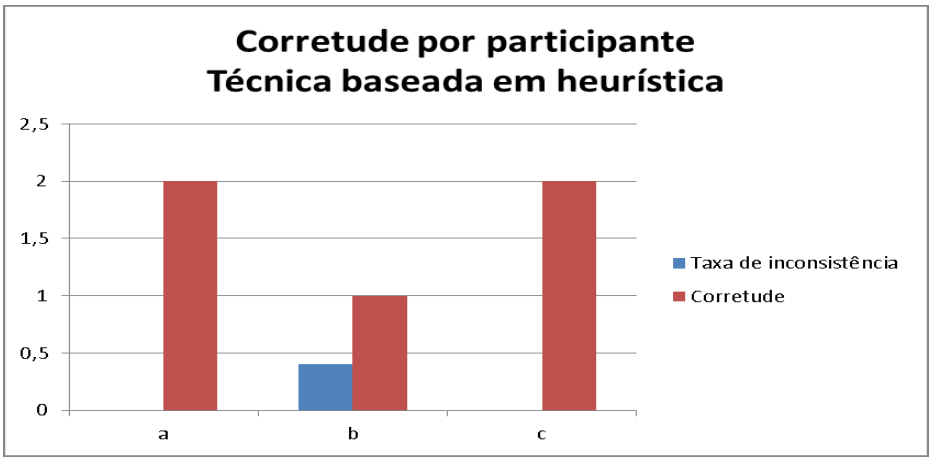
Figura 8 – Afetividade – Técnica baseada em heurística



Fonte: Elaborado pelo autor.

Na Figura 8, observa-se um padrão entre os indicadores de afetividade, visto que dois participantes obtiveram indicadores bastante similares. Já um terceiro participante alcançou valores aproximado dos demais, mas destoante do padrão apresentado. Devido a isso, torna-se interessante avaliar a qualidade dos modelos compostos de cada participante relacionando com seus indicadores de afetividade. Isto pode trazer indicios da influencia afetiva na composição de modelos. Na Figura 9, percebe-se que o participante (b), que possui indíces de frustração, excitação no período e excitação instantanea diferente dos outros participantes é justamente o que apresenta menor quantidade de modelos compostos corretamente e maior taxa de inconsistencia, para composições realizadas com técnica baseada em heurística.

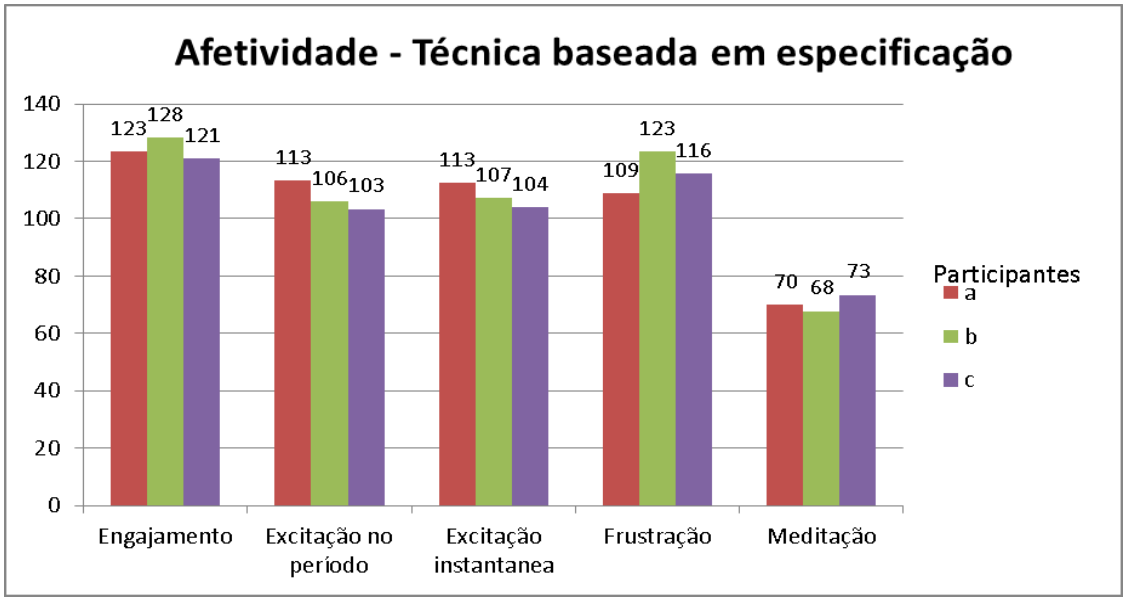
Figura 9 – Corretude por participante



Fonte: Elaborado pelo autor.

Com o intuito de realizar análise semelhante para técnica baseada em especificação, cabe também detalhar os mesmos indicadores. Na Figura 10, são apresentados os indicadorese afetivos de cada participante durante a composição dos modelos utilizando a técnica baseada em especificação.

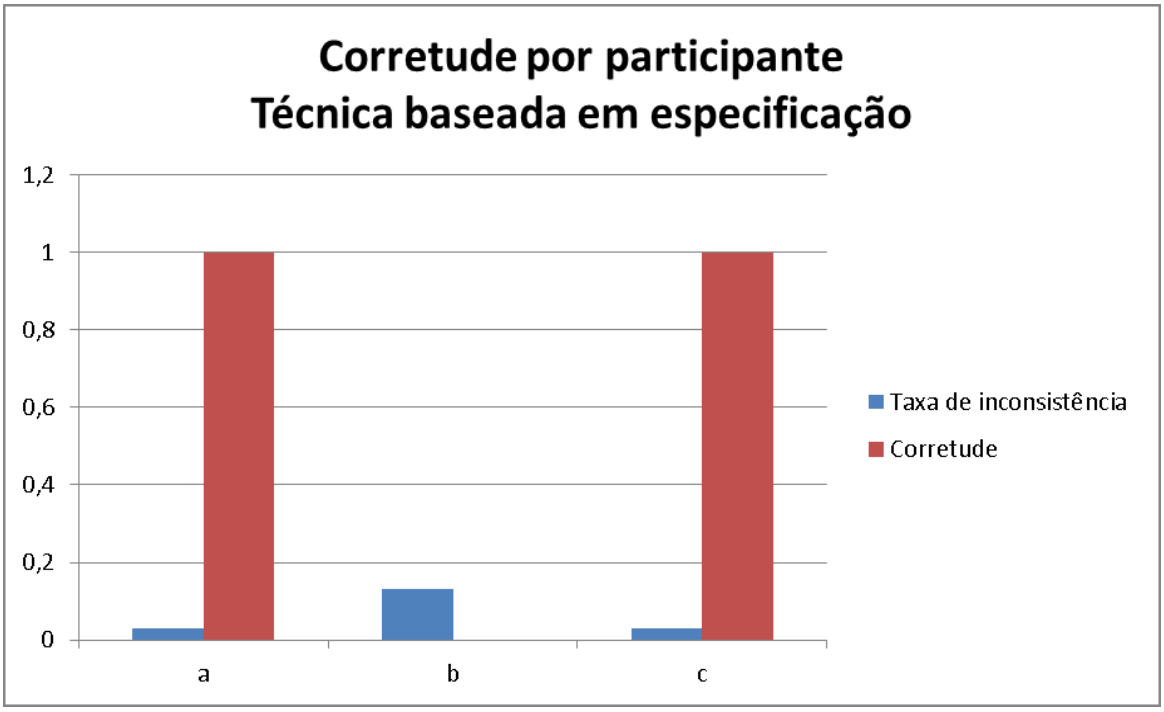
Figura 10 – Afetividade – Técnica baseada em especificação



Fonte: elaborado pelo autor.

Diferente dos resultados visualizado na Figura 8, na Figura 10 não é observada uma tendencia guiada por parte dos participantes. No entanto, o participante (b) ainda que tenha reduzido seu indice de frustração, manteve mais elevado que os demais. Onde o participante (c) manteve seu indice e o participante (a) reduziu. Para relacionar com os resultados referentes a qualidade dos modelos compostos usando a técnica baseada em especificação, detalha-se na Figura 11 os participantes com seus resultados.

Figura 11 – Corretude por participante

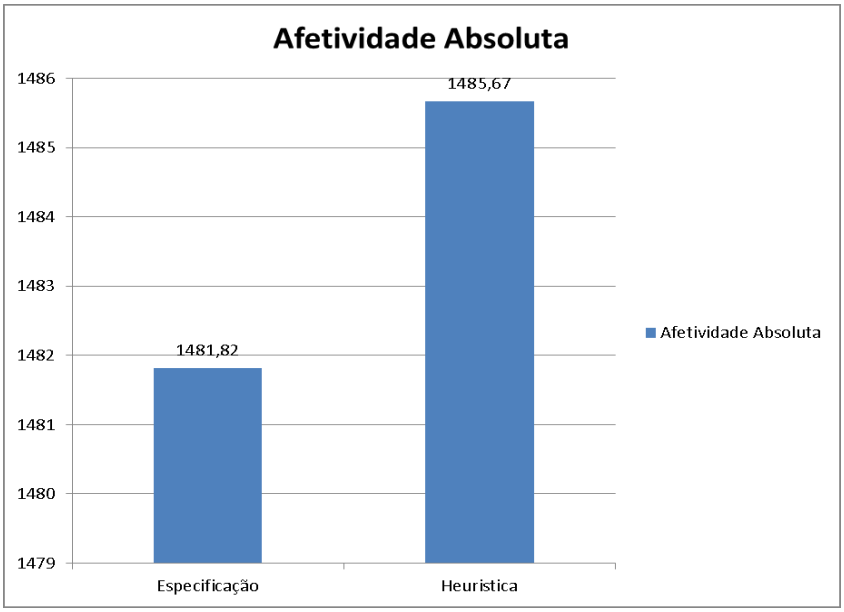


Fonte: Elaborado pelo autor.

Novamente destaca-se o participante (b) por não ter produzido nenhum modelo corretamente com a técnica baseada em especificação, além de também apresentar a maior taxa de inconsistência. Nota-se também que dos modelos propostos, os participantes (a) e (c) foram capazes de compor corretamente apenas 2 modelos, contudo, com taxa de consistencia inferiores as apresentadas pelo participante (b).

Devido aos resultados apresentados pelo participante (b) questiona-se a influência do indicador de frustração na qualidade dos modelos compostos. Ou ainda, se a relação entre altos índices de frustração e engajamento e, baixos de excitação não seriam um reflexo de que o participante (b) encontrava-se sob pressão. Pressão essa que pode ter as mais diferentes causas, como competitividade pessoal, tempo disponível para execução do experimento, pressão social, entre outras. Acredita-se que ainda é necessário realizar novos estudos para aprofundar os conhecimentos, compreender e traduzir em resultados e avaliações a influência da afetividade dos participantes na qualidade dos modelos compostos. Ainda sobre a hipótese H3, convem mencionar os resultados absolutos obtidos pelo indicador de afetividade, que totaliza os demais subindicadores em um, a afetividade (Figura 12).

Figura 12 – Afetividade absoluta



Fonte: Elaborado pelo autor.

De acordo com a Figura 12, é possível observa que a técnica de composição baseada em especificação teve um menor impacto na afetividade dos desenvolvedores em comparação a técnica baseada em heurística. Logo, os dados sugerem que a hipótese H3 nula ($H3_{1\text{-}0}$) é confirmada, uma vez que a técnica baseada em especificação apresentou menor impacto afetivo nos desenvolvedores.

Embora a técnica baseada em especificação exija um maior esforço de composição e produza uma maior taxa de inconsistência, ela apresentou um menor impacto na variável afetividade dos desenvolvedores. É bom lembrar que o indicador de afetividade é formado por vários outros indicadores, como engajamento, frustração, excitação no período e instantânea, e meditação. Contudo, ainda como estudo preliminar é possível verificar que os dados sugerem que os participantes tiveram um maior engajamento e uma menor frustração utilizando as técnicas heurísticas.

5 CONCLUSÃO

Uma vez que a atividade de compor modelos de software está presente nos processos de desenvolvimento de software colaborativo, este que é dinâmico pela sua natureza, passa a oportunizar experiências adversas no dia-a-dia dos desenvolvedores. Logo, abre-se espaço para as discussões quanto à eficiência e

eficácia das diferentes técnicas empregadas nesta atividade. Sendo assim, fez-se necessário buscar por respostas às incertezas evidenciadas no cotidiano dos desenvolvedores através de estudos empíricos.

Dados os resultados observados em relação ao esforço investido, verificou-se que para o uso da técnica baseada em heurística foi investido significativamente menos tempo do que com a técnica baseada em especificação para todos os participantes. O esforço investido com a técnica baseada em heurística chegou a representar para o participante (c) apenas 26% do esforço que foi investido com a técnica baseada em especificação. Para outro participante foi observado 72% do tempo em relação a técnica baseada em especificação. Também foi possível constatar que na fase 1 (Figura 3) é empregado muito mais tempo do que na fase 2. Desta forma, com a técnica baseada em heurística se verificou que na fase 1 foi investido entre 65% e 75% de todo o esforço do processo de composição de modelos de software. Para a técnica baseada em especificação foi verificado para todos os participantes, a aplicação da mesma demandou aproximadamente 65% do tempo de todo o processo de composição.

Com relação ao quão correto os modelos foram gerados também se observou melhores resultados com o uso da técnica baseada em heurística, que apresentou 5 modelos corretamente compostos contra apenas 2 obtidos através de especificação. A taxa de inconsistência também apresentou-se menor, sendo de apenas 4% para técnica baseada em heurística e de 19% para a técnica baseada em especificação.

Em respeito ao impacto afetivo das técnicas sob o desenvolvedor observou-se de maneira geral que a técnica baseada em heurística teve maior impacto, mas de forma muito sutil, com uma diferença de apenas 3,85 pontos. Ainda assim, conforme explicitado na seção de resultados, verificou-se a possibilidade de expandir estudos conferindo a qualidade dos modelos compostos em relação ao “estado de espírito” do desenvolvedor.

Analisando as amostras consideradas, o presente estudo possui um número baixo de participantes, o que inviabilizou o uso de métodos estatísticos para testar as hipóteses apresentadas. Como trabalhos futuros, deseja-se replicar o estudo executado com o objetivo de coletar um número maior de dados, o que viabilizará o teste das hipóteses usando métodos estatísticos.

Alguns fatores como a deficiência do software de captação de dados do eletroencefalograma fizeram com que o estudo fosse adiado até que se produzisse

um aplicativo capaz de captar os dados de forma numérica. A ausência do software IBM RSA, que usa da técnica baseada em heurística através de um mecanismo semiautomático se deve a indisponibilidade da ferramenta na versão desejada e de forma economicamente viável, o que também fez com que o tivesse a execução do procedimento experimental adiado até que se optou por desconsiderar tal ferramenta na aplicação do experimento. A indisponibilidade de potenciais participantes, dada a individualidade dos compromissos de cada um impediu também o aumento da amostra.

Espera-se que o presente estudo sirva como ponto de partida para futuros estudos. Isto porque possui caráter inovador e se posiciona como piloto ao seu propósito de esclarecer as incertezas da área de composição de modelos de software. Também se busca expor a possibilidade de utilizar novos meios, para explorar o fator humano, no caso o desenvolvedor, através de novas ferramentas disponíveis no mercada e atualmente não tão distante economicamente dos centros de pesquisa.

EVALUATING THE EFFORT OF COMPOSING SOFTWARE DESIGN MODELS: A neuroscience based study

Abstract: Developers use model composition techniques to integrate software models produced in parallel. To do that, they use heuristic based technique (e.g. IBM RSA, Astah), and specification based technique (e.g. Epsilon). The task of composing design models is not trivial, since developers have to integrate the common parts of the models and resolve the conflicts between involved models. If conflicts are not resolved appropriately, then opens up space for the appearance of inconsistencies. Thus, it is necessary to invest more effort to detect and resolve the generated inconsistencies. If not resolved, such inconsistencies can degrade the software architecture. The problem is there are no significant information about the impact of model composition techniques on composition effort and over brain activity of developers in this process. If a technique requires much effort and takes a developer to a unfavorable brain activity condition (i.e. a high degree of frustration), then it is questionable the adoption of this technique in real projects in the industry. Therefore, this study focuses on presenting a detailed experimental process beyond producing empirical evidences about the impact of such techniques in the composition effort and the brain activity of developers. A controlled experiment was conducted to measure the composition effort to integrate software models and to understand the brain activity responses when composing them. To do that, a non-invasive brain-computer interface was used.

Keywords: Model composition effort. Composing models techniques. Neuroscience.

NOTA EXPLICATIVA

Não houve notas explicativas.

REFERÊNCIAS

BASILI, V.; CALDIERA, G.; ROMBACH, H. **The Goal Question Metric Paradigm**. Encyclopedia of Software Engineering, vol. 2, p. 528–532, John Wiley and Sons, 1994.

CLARKE, S. **Composition of Object-Oriented Software Design Models**. Tese de Doutorado. Dublin City University, Janeiro,2001.

DOBING, B.; PARSONS, J. **How UML is used**, Communications of the ACM, vol. 49, no. 5, pp. 109-113, 2006.

EMOTIV: User Manual. Disponível em: <http://emotiv.com/developer/SDK/UserManual.pdf>. Acesso em: 26 de Maio de 2014.

EPSILON Software. Disponivel em: <http://www.eclipse.org/gmt/epsilon/>. Acesso em: 26 de Maio de 2014.

FARIAS, K.; GARCIA, A.;WHITTLE, J.; CHAVEZ, C.; LUCENA, C. **Evaluating the Effort of Composing Design Models**: A Controlled Experiment, In: Proceedings of the 15thInternational Conference on Model-Driven Engineering Languages and Systems (MODELS'12),Vol. 7590, pages 676-691, Innsbruck, Austria, 2012.

FRANCE, R., RUMPE, B. **Model-driven development of complex software:** a research roadmap. In: Future of Software Engineering at ICSE'07, pp. 37–54. 2007.

IBM Rational Software Architect. Disponivel em: <http://www.ibm.com/developerworks/rational/products/rsa/>. Acesso em: 26 de Maio de 2014.

KLEIN, J., HÉLOUËT, L., JÉZÉQUEL, J. **Semantic-based weaving of scenarios**. In: 5th Aspect-Oriented Software Development (AOSD'06), pp. 27-38, Bonn, Germany, Março de 2006.

KOLOVOS, D.; ROSE, L.; PAIGE, R. **The Epsilon book**, Disponível em: <http://www.eclipse.org/epsilon>, Acesso em: 26 de Maio de 2014.

KOMPOSE: A Generic Model Composition Tool. Disponível em: <http://www.kermeta.org/kompose>, 2011. Acesso em: 26 de Maio de 2014.

LANGE, C., CHAUDRON, M. **Effects of Defects in UML Models:** An Experimental Investigation. In: International Conference on Software Engineering (ICSE'06), pp. 401-410, China. 2006.

MENS, T. **A state-of-the-art survey on software merging**. IEEE Transaction on Software Engineering, pp. 449-462, vol. 28, no.5, 2002

NORRIS, N.; LETKEMAN, K. **Governing and Managing Enterprise Models**: Part 1. Introduction and concepts, IBM Developer Works. Disponível em: <www.ibm.com/developerworks/rational/library/09/0113_letkeman-norris> Acesso em: 26 de Maio de 2014.

OLIVEIRA, K. **Composição de UML Profiles**. Tese de Mestrado, FACIN/PUCRS, Porto Alegre, Brasil. 2008.

PETRE, M. **UML in practice**. In: 35th International Conference on Software Engineering (ICSE 2013), pp. 18-26, May 2013, San Francisco, CA, USA. 2013.

RUMBAUGH, J., JACOBSON, I., BOOCH, G. **The Unified Modeling Language Reference Manual**,Second Edition, 2004.

SHAO, D., KHURSHID, S., PERRY, D. SCA. **A Semantic Conflict Analyzer for Parallel Changes**, In: 7th European Software Engineering Conference/ACM SIGSOFT Symposium on the FSE, Amsterdam, Holanda, pp. 291-292, 2009.

THAKER, S., BATORY, D., KITCHIN, D., COOK, W. **Safe Composition of Product Lines**, In: 6th International Conference on Generative Programming and Component Engineering (GPCE'07), pp. 95–104, Salzburg, Austria, 2007.

WHITEHEAD, J. **Collaboration in Software Engineering**: A Roadmap, In: Future of Software Engineering at ICSE'07, pp. 214-225, 2007.

WHITTLE, J.; JAYARAMAN, P. **Synthesizing hierarchical state machines from expressive scenario descriptions**, ACM Transactions on Software Engineering and Methodology, vol. 19, no. 3, Janeiro 2010.

WOHLIN, C. **Experimentation in software engineering**: an introduction. Kluwer Academic Publishers, Norwell, USA, 2000.

APÊNDICE A – AffectiveConsoleApp: captação de dados afetivos

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;
using Emotiv;
```

```
namespace AffectiveConsoleApp
{
    class Writter
    {
        private static Writter instance;
        private TextWriter file;
        private Writter() { }

        public static Writter Instance
        {
            get
            {
                if (instance == null)
                {
                    instance = new Writter();
                }
                return instance;
            }

        }

        public void WriteHeader()
        {
            string  header  =  "Time\t\t\t"  +  "EngBor\t\t"  +
"ExcLong\t\t" + "ExcShrt\t\t" + "Frust\t\t" + "Medit";

            file.WriteLine(header);
        }

        public     void     WriteData(string     time,     float
esAf_EngBoredom,       float       esAf_ExcLongTerm,       float
esAf_ExcShortTerm,      float      esAf_Frustration,      float
esAf_Meditation)
        {
            string dataLine =
                time + "\t\t\t" +
                esAf_EngBoredom + "\t\t" +
                esAf_ExcLongTerm + "\t\t" +
                esAf_ExcShortTerm + "\t\t" +
                esAf_Frustration + "\t\t" +
                esAf_Meditation;

            file.WriteLine(dataLine);

        }

        public void open(String userFileName)
        {
            file = new StreamWriter(userFileName, false);
        }

        public void close()
        {
            file.Close();
        }
    }
}
```

```
//----------
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Emotiv;
using System.IO;
using System.Threading;

namespace AffectiveConsoleApp
{
    class EEG_Logger
    {

        EmoEngine engine; // Access to the EDK is viaa the
EmoEngine
        int userID = -1; // userID is used to uniquely
identify a user's headset

        EEG_Logger(String userFileName)
        {
            //get instance of Writter class
            Writter wt = Writter.Instance;
            //Set the filename as the user name
            wt.open(userFileName + ".txt");

            // create the engine
            engine = EmoEngine.Instance;
            engine.UserAdded                +=                new
EmoEngine.UserAddedEventHandler(engine_UserAdded_Event);
            engine.AffectivEmoStateUpdated        +=        new
EmoEngine.AffectivEmoStateUpdatedEventHandler(engine_AffectUpd
ated_Event);


            // connect to Emoengine.
            engine.Connect();

            // create a header for our output file
            wt.WriteHeader();
        }
        void       engine_UserAdded_Event(object       sender,
EmoEngineEventArgs e)
        {



            // record the user
            userID = (int)e.userId;

            Console.WriteLine("User Added Event has occured!
UserId:" + userID );
            // enable data aquisition for this user.
            engine.DataAcquisitionEnable((uint)userID, true);

            // ask for up to 1 second of buffered data
```

```
            engine.EE_DataSetBufferSizeInSec(1);

        }
        void     engine_AffectUpdated_Event(object     sender,
EmoStateUpdatedEventArgs e)
        {
            Writer wt = Writer.Instance;
            lock (e.emoState)
            {
                String                  time                  =
DateTime.Now.ToString("hh:mm:ss.fff tt");
                wt.WriteData(
                    time,

e.emoState.AffectivGetEngagementBoredomScore(),

e.emoState.AffectivGetExcitementLongTermScore(),

e.emoState.AffectivGetExcitementShortTermScore(),
                    e.emoState.AffectivGetFrustrationScore(),
                    e.emoState.AffectivGetMeditationScore()
                    );
            }
        }
        void Run()
        {
            // Handle any waiting events
            engine.ProcessEvents();

            //  If  the  user  has  not  yet  connected,  do  not
proceed
            if ((int)userID == -1)
                return;

        }

        static void Main(string[] args)
        {
            ConsoleKeyInfo key;
            Console.WriteLine("EEG Data Reader Example, insira
o nome do usuario:");
            EEG_Logger          eeglog          =          new
EEG_Logger(Console.ReadLine());

            while(true)
            {
                //to quit the LOOP press Q
/*                if (Console.KeyAvailable)
                {
                    key = Console.ReadKey(true);
                    switch (key.Key)
                    {
                        case ConsoleKey.Q:
                            break;
                        default:
                            continue;
                    }
                }
```

```
 */
                eeglog.Run();
                Thread.Sleep(100);
            }
            Writter wt = Writter.Instance;
            wt.close();
        }
    }
}
```