PONTIFÍCIA UNIVERSIDADE CATÓLICA
DO RIO DE JANEIRO

Juliana Soares Jansen Ferreira

Comunicação através de modelos no contexto do desenvolvimento de Software

Tese de Doutorado

Tese apresentada ao Programa de Pós-graduação em Informática do Departamento de Informática do Centro Técnico Científico da PUC-Rio, como requisito parcial para a obtenção do grau de Doutor em Informática.

Orientadora: Prof.ª. Clarisse Sieckenius de Souza

Rio de Janeiro
Abril de 2015

Juliana Soares Jansen Ferreira

Comunicação através de modelos no contexto do desenvolvimento de Software

Tese apresentada ao Programa de Pós-graduação em Informática do Departamento de Informática do Centro Técnico Científico da PUC-Rio, como requisito parcial para a obtenção do grau de Doutor em Informática. Aprovada pela Comissão Examinadora abaixo assinada.

Prof.ª. Clarisse Sieckenius de Souza
Orientador
Departamento de Informática – PUC-Rio

Profª. Tayana Uchôa Conte
UFAM

Prof. Rogerio Abreu de Paula
IBM Research - Brazil

Prof. Renato Fontoura de Gusmão Cerqueira
IBM Research - Brazil

Profª. Simone Diniz Junqueira Barbosa
Departamento de Informática – PUC-Rio

Prof. Carlos José Pereira de Lucena
Departamento de Informática – PUC-Rio

Prof. José Eugenio Leal
Coordenador Setorial do Centro Técnico Científico – PUC-Rio

Rio de Janeiro, 10 de abril de 2015

Juliana Soares Jansen Ferreira

Graduou-se em Bacharelado em Informática pela Pontifícia Universidade Católica do Rio de Janeiro (2007). Possui mestrado em Informática pela Universidade Federal do Estado do Rio de Janeiro (2010).

Ficha Catalográfica

Ferreira, Juliana Soares Jansen

Comunicação através de modelos no contexto do desenvolvimento de software / Juliana Soares Jansen Ferreira ; orientador: Clarisse Sieckenius de Souza. – 2015.
200 f. : il. ; 30 cm

Tese (dissertação)–Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2015.
Inclui bibliografia

1. Informática – Teses. 2. Interação humano-computador. 3. Engenharia semiótica. 4. Ferramenta de desenvolvimento de software. 5. Modelos de software. 6. Notação. 7. Cognitive dimensions of notations. 8. Comunicabilidade. I. Souza, Clarisse Sieckenius de. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Aos meus pais, Luiz e Angela, pelo amor e apoio.

Agradecimentos

Primeiramente, agradeço ao meu anjo da guarda, preposto de Deus no meu dia a dia, por toda proteção, saúde e inspiração sempre. Agradeço imensamente aos meus pais que sempre apoiaram minhas decisões, mesmo estas parecendo um pouco "kamikazes" ou totalmente sem explicação. Sem vocês não seria ninguém. A confiança, fé e amor que vocês têm por mim, é o que me faz acreditar que eu posso fazer sempre mais e melhor. Obrigada por todo apoio, amor, paciência e colo ao longo desta jornada. Amo vocês! Um agradecimento enorme a toda minha família e amigos que entenderam minhas ausências e me apoiaram nos momentos difíceis. Aos meus afilhados Miguel e Eduardo, meu agradecimento mais carinhoso e cheio de amor pelos sorrisos e beijos revigorantes. Agradeço às minhas queridas amigas Andrea, Vanessa e Priscila pela parceria, pela amizade, pelos conselhos e pelo exemplo de sucesso acadêmico e profissional que elas são. Meu imenso agradecimento à Camila e Tatiana, amigas queridas que o doutorado me deu de presente. Jamais teria sobrevivido ao primeiro ano sem vocês. Agradeço à Simone Barbosa, minha primeira professora de IHC, que sempre está de portas abertas pra qualquer coisa. Também agradeço à Carla Leitão que foi uma das responsáveis pela minha paixão pela pesquisa qualitativa. Agradeço à Luciana Salgado que me ajudou a navegar no "universo do doutorado", sendo um belo exemplo de professora, pesquisadora e também uma querida amiga. Um grande obrigado aos companheiros de SERG, principalmente Marcelle, Ingrid, Cleyton, Rafael e Luiz. Trabalhamos muito, mas nos divertimos quase na mesma proporção. Agradeço ao Renato Cerqueira pela co-orientação neste trabalho e pelas sempre ricas discussões que temos sobre a pesquisa. À minha querida orientadora Clarisse de Souza, não sei como agradecer a oportunidade e honra de ter sido sua orientada. Obrigada pelo apoio, pela paciência, pelo carinho e atenção desde a graduação até hoje. A sua entrada na minha vida realmente mudou o rumo das coisas e me fez ver caminhos que nunca havia imaginado. Por fim, agradeço

aos professores do DI da PUC-Rio que contribuíram para minha formação. Além disso, agradeço ao incentivo financeiro da CAPES.

Resumo

Ferreira, Juliana Soares Jansen; de Souza, Clarisse Sieckenius. **Comunicação através de modelos no contexto do desenvolvimento de Software.** Rio de Janeiro, 2015. 200p. Tese de Doutorado - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Desenvolvimento de software é um processo altamente colaborativo, no qual a construção do software é o objetivo comum. É apoiado em várias fases por ferramentas computacionais, dentre elas as ferramentas de modelagem de software. Modelos são parte importante do processo de desenvolvimento de software e o foco desta pesquisa, que tem como objetivo investigar a 'comunicabilidade' de modelos de software que são produzidos e consumidos através de ferramentas de modelagem. A comunicabilidade de modelos de software é a capacidade que estes artefatos têm de efetuar o processo de comunicação entre pessoas, ou a de serem usados como instrumentos para realizar parte significativa deste processo. As ferramentas de modelagem têm impacto direto nessa comunicabilidade, já que os produtores e consumidores de modelos interagem com tais ferramentas ao longo do processo de desenvolvimento do software. Durante essa interação, os modelos de software, que são artefatos intelectuais, são criados, alterados, evoluídos, transformados e compartilhados pelas pessoas envolvidas nas atividades de especificação, análise, design e implementação do software em desenvolvimento. Além da influência das ferramentas, a modelagem de software também deve considerar a utilização de notações previamente definidas como premissas para as atividades de modelagem. Esta pesquisa é uma investigação como ferramentas e notações de modelagem influenciam e apoiam o processo intelectual de produção e consumo de modelos de software. Temos a Engenharia Semiótica como teoria guia desta pesquisa, tendo em conta um aspecto essencial para esta que é: um estudo criterioso das ferramentas que os envolvidos no desenvolvimento do software utilizam para construir, usar e divulgar modelos através dos quais coordenam o seu trabalho de equipe. O uso de modelos no processo de desenvolvimento de software é um fenômeno que apresenta vários fatores que não podem ser isolados. Portanto, propomos a tripla *Tool-Notation-People* (TNP) como um recurso de articulação

para caracterizar questões observadas sobre modelos no desenvolvimento de software, ao longo de toda a pesquisa. Junto com a tripla TNP, apresentamos um método que combina as perspectivas cognitiva e semiótica para avaliar as ferramentas de modelagem de software, produzindo dados sobre a metacomunicação *designer*-usuário, sendo os usuários, neste caso, os desenvolvedores de software. Nosso objetivo é rastrear potenciais relações entre a experiência de interação humano-computador dos evolvidos no processo de desenvolvimento de software no momento de criar/ler/editar modelos com: (a) o produto (tipos de modelo) gerado neste processo; e (b) as interpretações que tais modelos evocam quando usados efetivamente em situações práticas do cotidiano para 'comunicar e expressar ideias e entendimentos'. A Engenharia Semiótica apresenta duplo interesse nesta pesquisa. Por um lado, como uma 'lente de observação', ela nos oferece diversos recursos para investigar e compreender a construção e uso de artefatos computacionais, seus significados e seus papéis no processo de comunicação. Por outro lado, um melhor entendimento sobre o processo completo que resulta, em última análise, na experiência do usuário durante a interação com o software é relevante para a evolução da própria teoria. Ou seja, esta pesquisa produziu mais conhecimento sobre as condições de comunicação e mútuo entendimento daqueles que, segundo a teoria, 'comunicam sua intenção e princípios de *design* através da interface', uma fonte potencialmente valiosa de explicações sobre problemas de comunicabilidade em IHC.

Palavras-chave

Interação Humano-Computador; Engenharia Semiótica; Ferramentas de desenvolvimento de software; Modelos de software; Notação; C*ognitive dimensions of notations*; Usabilidade, Comunicabilidade.

Abstract

Ferreira, Juliana Soares Jansen; de Souza, Clarisse Sieckenius (advisor). **Communication through models in the context of software development**. Rio de Janeiro, 2015. 200p. Ph.D Thesis - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Software development is a highly collaborative process where software construction is the common goal. It is supported at several stages by computer tools, including software modeling tools. Models are important artifacts of the software development process and constitute the focus of this research, which aims to investigate the 'communicability' of software models produced and consumed with the support of modeling tools. Software model communicability is the capacity that such artifacts have of carrying and effecting a communication process among people, or of being used as an instrument to perform a significant part of such process. Modeling tools have a direct impact in that communicability, since model's producers and consumers interact with those tools throughout the software development process. During that interaction, software models, which are intellectual artifacts, are created, changed, evolved, transformed and shared by people involved in activities of specification, analysis, design and implementation of the software under development. Besides the influence of tools, software modeling also needs to take into consideration previously defined notations as premises for modeling activities. This research is an investigation on how tools and notations influence and support the intellectual process of production and consumption of software models. We have Semiotic Engineering as our guiding theory given the essence of it that is: a careful study of tools people interact with to build, use and publish models through which they coordinate teamwork. The use of models in the software development process is a phenomenon that includes several factors that cannot be isolated from each other. Therefore, we propose a Tool-Notation-People triplet (TNP triplet) as a means of articulation to characterize observed issues about models in the software development. Along with the TNP triplet, we introduce a method that combines semiotic and cognitive perspectives to evaluate software modeling tools, producing data about the emission of *designer*-user metacommunication, users in this case being software developers. We aim to track potential relations between the human-computer

interaction experience of those involved in the software development process while creating/reading/editing models with: (a) the product (types of models) generated in the process; and (b) the interpretations that such models evoke when used effectively in everyday practical situations to 'communicate and express ideas and understandings'. The interest of working with Semiotic Engineering in this research is twofold. First, as an 'observation lens', the theory offers many resources to investigate and understand the construction and use of computational artifacts, their meanings and roles in the communication process. Second, a better perspective about the complete process that results, ultimately, in the user experience during the interaction with the software is relevant for the theory's own evolution. In other words, this research has produced further knowledge about the communication conditions and mutual understanding of those who, according to the theory, 'communicate their intent and *design* principles through the interface', a potentially valuable source of explanations about communication problems in HCI.

Keywords

Human-computer interaction; Semiotic Engineering; Software development tools; Notations; Cognitive dimensions of notations; Usability; Communicability.

Sumário

Lista de figuras

Lista de tabelas

“Só sei que nada sei, e o fato de saber isso, me coloca em vantagem sobre aqueles que acham que sabem alguma coisa. ”

Sócrates

"The real voyage of discovery consists not in seeking new landscapes, but in having new eyes."

Marcel Proust

"I have no special talent. I am only passionately curious."

Albert Einstein

1
Introdução

Modelos são representações parciais ou simplificadas da realidade que apresentam um conjunto de objetos, com propriedades (*i.e.* relações, funções, operações associadas, etc.) precisas e consistentemente definidas. Modelos são abstrações de algum objeto ou aspecto do mundo real. A ideia de criar abstrações do mundo para entender questões complexas é inerente ao comportamento humano. Para investigar um fenômeno complexo, normalmente, são criados vários modelos. (Lieberman, 2006)

Um modelo deve apresentar três características principais: (1) característica de redução, o modelo reflete somente uma seleção relevante das propriedades originais do objeto modelado, de tal forma que se concentra em determinados aspectos de interesse sobre o objeto; (2) característica de mapeamento, o modelo é baseado em um objeto no mundo real e tomado como referência em relação a alguma propriedade deste objeto; e, por fim, (3) característica pragmática, o modelo deve substituir o objeto original, ou seja, ser usável, em algum aspecto ou propósito. (Stachowiak, 1973 *apud* Ludewig, 2003) (Stachowiak, 1973 *apud* Muller et al., 2011)

A investigação de modelos deve considerar e distinguir duas perspectivas de observação: perspectiva da **notação do modelo (Mnot)**, onde será considerada a notação de representação de objetos e propriedades associadas do modelo; e a **semântica do modelo (Msem)**, quando fazemos referência ao significado de uma visão do modelo representado por uma notação. Essas perspectivas estão diretamente relacionadas e podem se confundir na discussão sobre modelos. Portanto, optamos por fazer a distinção entre perspectivas do modelo usando os mnemônicos Mnot e Msem junto à palavra ‘modelo’ ao longo do texto desta tese. Quando a referência for relacionada às duas perspectivas, utilizaremos somente a palavra ‘modelo’.

Modelos são recursos que nos ajudam a resolver problemas. Através da modelagem do domínio do problema, podemos ter visões diferentes, considerar

partes ou o todo, simular diferentes cenários em busca da solução do problema a ser resolvido. (Funt, 1995) (Larkin e Simon, 1995) Através da interação com o modelo (Mnot), os envolvidos que buscam solucionar o problema, possuem um objeto que pode ser manipulado durante as discussões e experimentações para representar diferentes situações e soluções propostas que seguem fomentando as discussões dos envolvidos sobre as possíveis soluções. (Magnani, 2005) Além disso, com sua característica de abstrair detalhes, o modelo (Msem) colabora para o processo de raciocínio segmentado sobre as partes mais relevantes do problema.

O desenvolvimento de um software normalmente é motivado pela necessidade de solução de um determinado problema identificado no domínio de negócio. (Pressman, 2010) Portanto, para que o software desenvolvido alcance o objetivo de ajudar na solução de um problema, os *designers* do software precisam entender profundamente esse problema. Uma das ferramentas para apoiar o entendimento do problema são justamente os modelos de software.

Independente da metodologia de desenvolvimento de software, os modelos estão sempre presentes como artefatos produzidos em diversas etapas do processo. Seu uso é reconhecido e apresentado nas referências mais tradicionais da Engenharia de Software (Pressman, 2010) (Sommerville, 2010) e ocorre em praticamente todas as fases do desenvolvimento: da especificação de requisitos, com modelos de domínio e modelo de caso de uso, passando pela especificação de arquitetura, até o modelo físico do banco de dados do software final. (Pressman, 2010)

Os modelos de software apresentam diferentes níveis de abstração e visões sobre o software a ser desenvolvido. A semântica do modelo varia ao longo do processo de acordo com a necessidade da etapa de desenvolvimento. Existem diversas notações definidas como padrões e utilizadas para representar modelos de software. (Pressman, 2010) Portanto, temos no processo de produção e consumo de modelos de software, parte do processo de desenvolvimento, um contexto rico para a investigação tanto de aspectos semânticos quanto de aspectos de representação (notacionais) de modelos de software.

No processo de modelagem de software temos a UML (OMG, 2011) como notação mais conhecida e utilizada, apresentando diversos tipos de diagramas para representar visões diferentes do modelo de domínio do problema relacionado ao software. A ideia de utilizar uma linguagem de modelagem é ter um padrão, uma

base de referência para que os produtores e consumidores de modelos de software possam "falar a mesma língua" e alinhar seus entendimentos através dos modelos construídos. A UML é a linguagem central para modelagem de software, porém a combinação da UML com outras notações, já existentes ou criadas, é uma realidade identificada na prática do processo de desenvolvimento de software. (Petre, 2013b) A representação do modelo (Mnot) está diretamente relacionada com a intenção de uso do modelo e seus potenciais consumidores. O objetivo é que a notação do modelo (Mnot) seja capaz de viabilizar e facilitar a semântica (interpretação do significado) do modelo (Msem), conforme intencionada por seu produtor, para seus consumidores.

As ferramentas de modelagem de software têm um papel muito relevante neste contexto, principalmente quando o desenvolvimento é realizado por equipes grandes ou dispersas geograficamente, ou quando a complexidade do problema exige uma rastreabilidade de evolução mais rígida. Nem sempre são usadas ferramentas de modelagem de software para a construção de modelos, principalmente no estágio inicial do *design* do software (Petre e Van Der Hoek, 2013), porém a forma como os *designers* interagem, tomam decisões, se relacionam com os artefatos mesmo sem o uso de ferramenta, pode dar indicações importantes para que desenvolvedores de ferramentas de modelagem entendam o contexto do problema que tais ferramentas procuram solucionar: o próprio contexto da modelagem de software.

As abordagens *model-driven*[1] têm particular interesse tanto em relação à UML quanto às ferramentas de modelagem de software, já que o modelo é o centro de todo o desenvolvimento. A partir de um "bom" modelo que representa o domínio do problema, seguido de um processo de transformação desse modelo, o código-fonte do software é gerado. (Brambilla et al., 2012) Porém, a maioria das pesquisas relacionadas às abordagens *model-driven* estão concentradas na transformação do modelo (Mnot) e geração de código, deixando a parte do 'bom' modelo (Msem) como uma premissa, sem volume de pesquisas específicas. Se o modelo (Msem) do domínio do problema apresentar falhas, a transformação e a geração do código ficam comprometidas. Portanto, os produtores de modelos de

[1] Em português são abordagens orientadas a modelos, porém esta expressão é muito menos usada no domínio da tese do que a original, mesmo entre pesquisadores brasileiros, e por esta razão utilizamos a expressão em inglês no texto desta tese.

software devem ter um sólido conhecimento do contexto do problema que o software visa resolver ou amenizar, assim como da notação de modelagem (UML) para construir bons modelos (Msem). Esta questão já está sendo indicada como tópico de pesquisas necessárias para o sucesso das abordagens *model-driven*. Tais pesquisas também indicam a necessidade de buscar conhecimentos da área de Interação Humano-Computador (IHC) para a construção das ferramentas que dão suporte a esta abordagem. (Whittle et. al, 2014)

Segundo Winograd (1996), o *design* de qualquer artefato (ex. modelos de software, software em si), envolve tomadas de decisão do *designer* durante sua construção. Assim, uma vez construído, o artefato leva consigo a marca de intenção do *designer* sobre o que ele faz e como deve ser entendido e usado. Esta posição vai ao encontro da abordagem da Engenharia Semiótica (de Souza, 2005), já que ambas indicam a necessidade de fornecer ao *designer* o suporte necessário para a produção do artefato, especialmente no que diz respeito ao tratamento desta intenção de *design*.

A Engenharia Semiótica (de Souza, 2005) considera a interação humano-computador como um processo de comunicação entre humanos mediada por computador, onde temos três partes envolvidas: *designer*[2], usuário e software. A comunicação acontece entre *designer* e usuário através do software em tempo de interação. O software é o preposto do *designer* (Figura 1 retirada de PIMENTEL et al., 2011) e apresenta somente os caminhos de conversa que foram antecipados e projetados pelo *designer* durante o desenvolvimento do software. O software representa e age em nome do *designer* no processo de comunicação.

[2] Ao usar o termo *designer*, a Engenharia Semiótica de fato se refere a um indivíduo que seja porta-voz de toda uma equipe de *design* e desenvolvimento, cujo trabalho é o que leva à construção do software do qual a interface funciona como um representante e interlocutor na comunicação com o usuário.

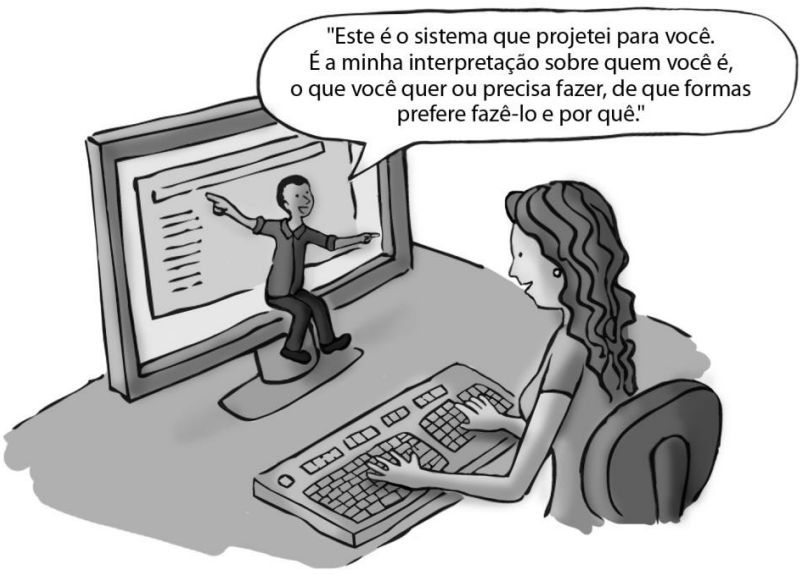


Figura 1. O preposto do *designer*

Durante a interação do usuário com o software, existe um processo de metacomunicação, a comunicação sobre como, quando, onde e por que se comunicar com o sistema idealizado pelo *designer*. Através da avaliação dessa metacomunicação *designer*-usuário através do software é caracterizada a comunicabilidade do software.

A Engenharia Semiótica se baseia na semiótica de Peirce (1992-1998) e de Eco (1976, 1983) onde o conceito central é o signo. Um signo é "algo que representa alguma coisa para alguém em alguma capacidade". Em estudos preliminares relacionados a modelos (Ferreira e de Souza, 2013) (Ferreira et al., 2014) identificamos algumas questões relacionadas à retórica especulativa, que é parte da teoria peirceana. A retórica especulativa lida com a existência dos signos depois que estes foram percebidos por uma mente ou inteligência (Santaella, 2004). Considera os signos quando estão em uso, quando inseridos em um contexto, sendo empregados no processo de semiose (*i.e.* de interpretação e significação) dos envolvidos. Essa é a situação que a Engenharia Semiótica foca, oferecendo ferramentas para investigar os signos em um determinado contexto com uma determinada intenção. Com o uso da Engenharia Semiótica, nós podemos enquadrar um espaço onde os signos são observados e investigados, podemos analisar como eles estão sendo emitidos e recebidos, durante a comunicação *designer*-usuário através de ferramentas e durante o seu processo de produção e consumo.

Nesta pesquisa investigamos o processo de produção e consumo de modelos de software, onde pessoas fazem uso de notações e ferramentas para comunicar aspectos relacionados ao software em construção através de modelos. Todos os envolvidos no processo de desenvolvimento têm como objetivo comum o software final. Durante a construção do software, modelos são produzidos buscando representar diversas visões do domínio relacionado ao software. Ao longo do processo de desenvolvimento tais modelos também são consumidos por outras pessoas que têm a semântica (Msem), e em alguns casos também a notação (Mnot), do modelo como insumo para outras atividades do desenvolvimento de software. Tais modelos precisam apresentar uma propriedade que, por extensão a um conceito original da Engenharia Semiótica (de Souza, 2005; de Souza e Leitão, 2009), chamamos de *comunicabilidade de modelos de software*, que é a capacidade que estes artefatos têm de efetuar o processo de comunicação entre pessoas, ou a de serem usados como instrumentos para realizar parte significativa deste processo.

Como assinalado acima, o conceito de comunicabilidade de modelos de software pode ser considerado um 'alargamento metafórico' do conceito de comunicabilidade da Engenharia Semiótica, considerando o potencial poder dos artefatos produzidos (modelos) em trazer consigo marcas da metacomunicação *designer*-usuário da ferramenta através da qual se interage (constrói, usa, evolui, etc.) com tal artefato.

Para conhecer mais sobre a relação da comunicabilidade de modelos com ferramentas e notações de modelagem, a estratégia desta pesquisa foi de investigar como a interação de desenvolvedores com ferramentas computacionais de modelagem de software pode agregar aos modelos criados (e posteriormente usados) características de comunicabilidade desejáveis ou indesejáveis. Por causa da estratégia escolhida, esta tese se posiciona na fronteira entre as áreas de IHC e de Engenharia de Software.

1.1
Contexto e motivação para a pesquisa

Nosso contexto de pesquisa é o processo de desenvolvimento de software onde pessoas colaboram e interagem para a construção de um software. Esta pesquisa teve como foco as atividades de modelagem de software realizadas com

o apoio de ferramentas, tendo como motivação o uso dos modelos como instrumentos de comunicação entre os envolvidos no desenvolvimento. Em uma equipe de desenvolvimento as pessoas se comunicam através de modelos que, assim, deveriam – para ser eficazes e eficientes neste processo – ter 'boa comunicabilidade'. O contexo de modelagem considerado envolve a produção e consumo de modelos, onde várias pessoas interagem de formas diferentes com modelos (Mnot) durante o processo de desenvolvimento de software com o apoio de ferramentas. Durante este processo de produção e consumo, os desenvolvedores são os usuários das ferramentas de modelagem, e como qualquer usuário de qualquer software, podem enfrentar problemas de IHC. (Figura 2)

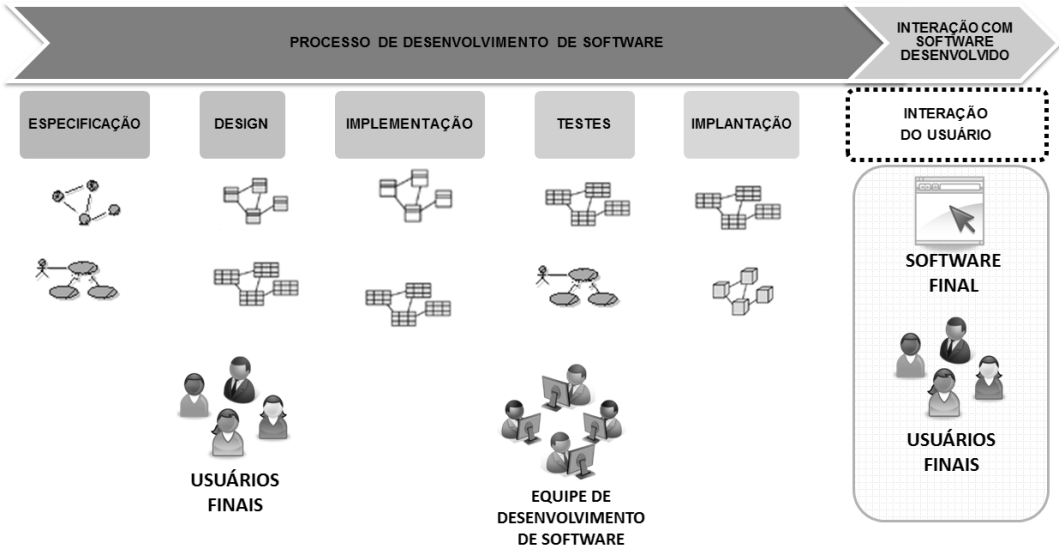


Figura 2. Modelos no processo de desenvolvimento

Considerando o processo de produção e consumo de modelos de software, que ocorre em diversas fases ao longo do processo de desenvolvimento, temos como motivação, o que podemos considerar como ***uma questão de pesquisa intrigante e a ser respondida apenas no longo prazo:***

Existe relação entre a experiência do desenvolvedor de software como usuário das ferramentas de modelagem, durante a produção e consumo de modelos, e a experiência do usuário final com o software construído por esse desenvolvedor?

Tendo a Engenharia Semiótica como teoria base, o que nos intriga é saber se possíveis rupturas da comunicação entre, por um lado, os *designer*s de ferramentas de apoio ao desenvolvimento de software e, por outro, os

desenvolvedores que as utilizam, identificadas durante o processo de desenvolvimento de um software, podem ser ‘propagadas’ para a experiência de IHC do usuário final deste mesmo software.

Tanto quanto é o nosso conhecimento, este tipo de pesquisa nunca foi realizado, embora seja perfeitamente plausível supor que: (1) problemas de IHC com ferramentas de desenvolvimento de software podem afetar negativamente a qualidade do software produzido no final; e (2) que algum (ou alguns) destes efeitos negativos esteja direta ou indiretamente ligado à experiência de interação do usuário final com o produto final do desenvolvimento.

Esta pesquisa foi motivada por essa conjectura sobre a propagação de problemas de IHC enfrentados por desenvolvedores de software para a experiência de IHC do usuário final do software. Se esta propagação existir, um enorme esforço de melhoria do *design* de interfaces e interação pode estar deixando de dar conta da origem do problema e, portanto, deixando de resolvê-lo definitivamente. E para garantir que a resposta é "não" (*i.e.* de que não existe propagação) é preciso de toda forma fazer a pesquisa.

Começamos portanto por investigar a experiência de IHC de desenvolvedores como usuários de ferramentas que apoiam o processo de desenvolvimento de software. Ao longo desse processo, desenvolvedores produzem diversos tipos de artefatos, utilizando diferentes tipos e naturezas de ferramentas.

1.2
Questão de pesquisa e objetivos

Considerando a questão em longo prazo, concentramos a pesquisa aqui apresentada em uma das partes do processo de desenvolvimento: a produção e consumo de modelos de software. Portanto, nosso contexto de pesquisa está concentrado na parte esquerda da Figura 3, onde temos o desenvolvedor de software interagindo com o *designer* da ferramenta de modelagem. Definido o nosso contexto de pesquisa, nossa questão de pesquisa é:

Como ferramentas e notações de modelagem influenciam e apoiam o processo intelectual de produção e consumo de modelos de software através da interação humano-computador?

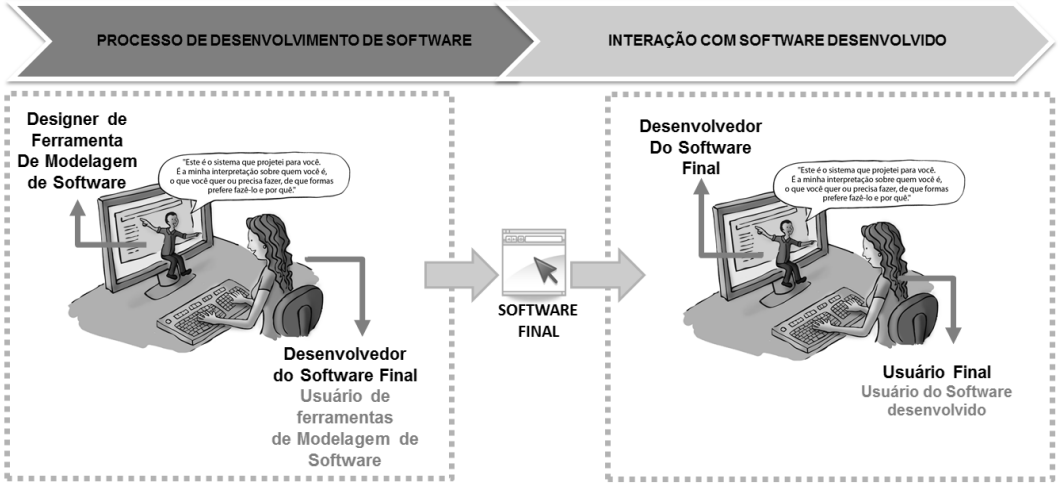


Figura 3. Contexto de pesquisa em longo prazo

Para investigarmos o contexto relacionado ao processo de produção e consumo de modelos, realizamos estudos para a investigação do contexto de desenvolvimento de software e para a investigação da interação de ferramentas de modelagem de software. Como ferramenta para estes estudos, desenvolvemos a **tripla t*ool-notation-people* (tripla TNP)**, que é um recurso de articulação de três fatores relacionados à modelagem de software: Ferramenta (*Tool*), Notação (*Notation*) e Pessoas (*People*). Com a tripla podemos caracterizar as questões e problemas identificados nas perspectivas do contexto de desenvolvimento de software e da interação com ferramentas de modelagem de software, relacionando-os a um ou mais fatores da tripla, bem como possíveis relações entre eles.

Na **investigação do contexto de desenvolvimento de software**, coletamos dados de profissionais com experiência em desenvolvimento de software, através de questionário e entrevista, sobre o processo de produção e consumo de modelos de software, considerando ferramentas, notação, intenção associada aos modelos, pessoas envolvidas e outras informações que nos possibilitaram identificar

categorias de problemas ou questões relevantes para o processo de produção e consumo de modelos de software. Nesta fase de investigação consideramos a experiência dos profissionais em relação à modelagem feita com e sem apoio de ferramentas. Ao conhecer contextos onde ferramentas não são usadas, podemos identificar as razões da ausência destas e, potencialmente, prover insumo sobre características ausentes ou problemáticas de ferramentas de modelagem já conhecidas pelos profissionais aos desenvolvedores de ferramentas de modelagem de software.

Na **investigação da interação de ferramentas de modelagem de software**, realizamos a avaliação de quatro ferramentas de modelagem de software, reconhecidas no mercado, utilizando um método de avaliação, também desenvolvido para esta pesquisa, chamado ***Combined Semiotic-Cognitive Evaluation method* (CSCE)**. O CSCE foi desenvolvido a partir da combinação de dois métodos de avaliação de IHC: Método de Inspeção Semiótica (MIS), método de avaliação da Engenharia Semiótica (de Souza e Leitão, 2009), e o "*Cognitive Dimensions of Notations framework*" (CDNf), método de análise de usabilidade de artefatos informacionais (Green e Blackwell, 1998). Junto com a tripla TNP, o CSCE possibilitou uma caracterização, mais abrangente que os métodos combinados isoladamente, da comunicabilidade de ferramentas de modelagem no contexto de desenvolvimento de software.

Os resultados das investigações do contexto de desenvolvimento de software e da interação com ferramentas de modelagem de software foram triangulados entre si e com achados de pesquisas anteriores, possibilitando a identificação de problemas e questões mais precisas sobre a experiência de IHC dos desenvolvedores no processo de produção e consumo de modelos de software e também a identificação de problemas e questões de ordem superior, considerando o contexto de desenvolvimento de software como um todo, onde pessoas colaboram para a construção de um mesmo artefato ao final: o software.

1.3 Contribuições

A principal contribuição desta pesquisa é a caracterização de problemas e questões identificadas no processo de produção e consumo de modelos de software com apoio de ferramentas de modelagem, considerando os fatores da

tripla TNP. Os achados estão relacionados com a representação dos modelos (Mnot), com o significado destes (Msem) e também com a relação entre semântica e notação dos modelos. Os achados relacionados à forma como notação e ferramentas influenciam e apoiam a produção e consumo de modelos de software são indicações que podem colaborar para o melhor entendimento e apropriação da notação (*i.e.* UML). Podem também ser usados como insumos interessantes para os *designer*s desta natureza de ferramentas (*i.e.* ferramentas que apoiam o desenvolvimento de software) e indicar caminhos para investigação de práticas de desenvolvimento de software que ainda são pouco pesquisadas de forma empírica. (Budgen et al., 2011) (Petre, 2013b)

Nossos achados apresentam alguns *insights* sobre a questão em longo prazo sobre a propagação de problemas de IHC enfrentados por desenvolvedores de software para a experiência de IHC do usuário final do software. O processo de produção e consumo de modelos de software oferece um contexto rico onde os fatores ferramentas, notações e pessoas estão fortemente relacionados. Porém a modelagem é somente uma parte do processo de desenvolvimento. Para que se encontre uma resposta sobre a propagação de problemas de IHC no desenvolvimento de software, a investigação deve ser ampliada para outras porções do processo de desenvolvimento, onde outras ferramentas e notações podem ser utilizadas. Nossos achados, então, podem ser usados como referência ou mesmo aprofundados em outras pesquisas, até por outras áreas de conhecimento além de IHC e Engenharia de Software.

Ter a Engenharia Semiótica como guia nos possibilitou realizar a pesquisa na área fronteiriça de IHC e Engenharia de Software, tendo como principal contribuição um *método para avaliação semiótico-cognitiva de interfaces de ferramentas de modelagem de software*, referenciadas ao papel comunicativo que modelos desempenham no processo de desenvolvimento de software. O método para avaliação semiótico-cognitiva de interfaces de ferramentas de modelagem de software é defindo por dois compomentes: a tripla TNP e o CSCE. A definição da tripla TNP como referência não só para os estudos, mas para a pesquisa como um todo, condiciona o pesquisador a sempre considerar os três fatores e suas relações e como estão influenciando, ou sendo influenciados, durante a modelagem de software no contexto do desenvolvimento. O CSCE não é um método de avaliação simples, que pode ser utilizado por qualquer pesquisador. É necessária

uma experiência sólida com métodos de avaliação semiótica, com as *cognitive dimensions of notations* e uma boa base de pesquisa qualitativa para realmente tirar proveito do método. De qualquer forma, mesmo não sendo um método *off-the-shelf*, esta pesquisa mostra, como será visto nos capítulos a seguir, como o seu uso pode render resultados inovadores e relevantes, abrindo nova perspectiva sobre a experiência de desenvolvedores como usuários enquanto constroem software destinado a terceiros. Portanto a relevância da pesquisa está em: (a) possibilitar uma apreciação em profundidade (e sob 2 aspectos distintos e complementares – semiótico e cognitivo) os efeitos do uso de certas ferramentas de modelagem em processos de desenvolvimento de software; e (b) informar *designer*s de interfaces de ferramentas de modelagem a respeito de traços fundamentalmente relevantes que elas poderão exibir, dependendo das escolhas feitas durante o processo de design de interação.

Esta pesquisa apresentou uma dinâmica interessante em relação ao ponto de observação da investigação. Considerando os fatores da tripla TNP, em alguns momentos investigamos pontos bem específicos da modelagem de software (ex. criação de novas representações usando o elemento nota da UML), e em outros momentos considerando o contexto de desenvolvimento como um todo (ex. quando um modelo será transformado em outro artefato ou disponibilizado para outras pessoas). Essas mudanças de perspectiva de observação, uma espécie de *zoom-in/zoom-out*, durante a execução da pesquisa e sempre usando a tripla TNP como referência, nos possibilitou considerar questões mais abrangentes que somente o momento de interação com a ferramenta. Pudemos observar como essa interação tem relação com etapas anteriores ou posteriores do processo relativamente à que está em foco de observação no momento. Sendo nosso contexto de pesquisa um processo (de desenvolvimento de software), essa dinâmica no ponto de observação rendeu achados bastante interessantes.

E por fim, esta pesquisa também apresentou como contribuição indicações de um interessante caminho de evolução para a pesquisa da teoria de Engenharia Semiótica, possibilitando vislumbrar os efeitos de um potencial ‘alargamento metafórico’ do conceito original de comunicabilidade e alcançar um melhor entendimento sobre o processo completo que resulta, em última análise, na experiência do usuário durante a interação com o software. É possível que a identificação de questões de IHC experimentadas por profissionais durante o

processo de desenvolvimento de software possa oferecer explicações sobre problemas de comunicabilidade que seu produto possa causar na interação com usuários finais.

1.4 Organização do texto

Procurando apoiar a leitura desta tese, evitando ambiguidades e redundâncias, definimos alguns termos que devem ser conhecidos pelo leitor. O termo **ferramenta** foi utilizado quando fazemos referência a um software que apoia alguma atividade do processo de desenvolvimento (*i.e.* ferramentas de modelagem) e o termo **software** para referenciar o que é construído no processo de desenvolvimento, do qual a modelagem faz parte.

O texto está estruturado da seguinte forma: no capítulo 2 apresentamos os trabalhos relacionados à nossa pesquisa, utilizando por mera questão de organização, os três elementos da tripla TNP para agrupar e discutir esses trabalhos. No capítulo 3 discutimos o ferramental metodológico desenvolvido, detalhando a tripla TNP e o CSCE, duas ferramentas que são a parte central da pesquisa empírica que fundamenta esta tese. No capítulo 4 apresentamos a metodologia da pesquisa empírica que realizamos, detalhado suas fases, insumos e produtos. No capítulo 5 apresentamos os estudos propriamente ditos, seus resultados e discussão sobre esses resultados encontrados. E, finalmente, no capítulo 6, fazemos as considerações finais, destacando em maiores detalhes as contribuições e indicando trabalhos futuros que já vislumbramos a partir do que pudemos avançar na pesquisa que fizemos.

2
Trabalhos Relacionados

Neste capítulo apresentamos e discutimos o conteúdo de trabalhos relacionados a esta pesquisa. Primeiramente, discutimos o papel e relevância dos modelos no processo de desenvolvimento de software, considerando a UML como a principal notação de modelagem. Também apresentamos pesquisas realizadas considerando alguns dos fatores da tripla TNP, ou uma combinação deles. Elas foram parte da nossa motivação e inspiração para combinar FERRAMENTA, NOTAÇÃO e PESSOAS em uma estrutura única para articular uma abrangente caracterização dos fenômenos que ocorrem durante o processo de desenvolvimento de software, particularmente relacionados à modelagem.

2.1
Modelos e Desenvolvimento de Software

Um modelo pode ser definido como a abstração de um sistema complexo e tem como propósitos o raciocínio, simulação, análise ou comunicação de detalhes específicos de um determinado assunto ou domínio. (Lieberman, 2006) Um modelo é um conjunto de assertivas, operações, propriedades relativas a (algum aspecto de) algum objeto. Representações de abstrações e definições técnicas através de modelos de software são partes significativas da prática da Engenharia de Software. (Pressman, 2010)

O processo de desenvolvimento de software e os modelos gerados ao longo deste processo são parte da mesma história (Ludewig, 2003) (Sommerville, 2003) (Pressman, 2010). A UML – *The Unified Modeling Language*™ (Rumbaugh et al., 1999) (OMG, 2011), notação de modelagem mais utilizada no desenvolvimento de software, é uma linguagem de modelagem para especificação, visualização, construção e documentação de artefatos de software. Ela foi definida a partir do Processo Unificado de Desenvolvimento de Software (Rumbaugh et al., 1999). Esse processo unificado foi originalmente construído a partir das

melhores características dos processos tradicionais de desenvolvimento de software da época, porém vem se atualizando junto com as novas especificações da UML, inclusive levando em consideração as metodologias ágeis de desenvolvimento de software. (Jacobson, 2002) (Ambler, 2002) (Rumpe, 2004) (Pressman, 2010)

A investigação de modelos pode ser focada na representação do modelo (Mnot) ou na semântica do modelo (Msem). A UML (OMG, 2011), notação selecionada para o contexto desta pesquisa, apresenta um conjunto de vários diagramas que podem ser combinados durante a construção do software, procurando representar e transmitir significado em diversos níveis. Portanto, a representação visual através de diagramas UML (Mnot) faz parte dos modelos investigados nesta pesquisa.

Algumas pesquisas levantaram dados junto aos profissionais de desenvolvimento de software onde alguns diagramas da UML foram indicados como os mais utilizados. (Dobing e Parsons, 2006) (Erickson e Siau, 2007) (Petre, 2013b) (Reggio et al., 2014) Porém, dados a abrangência e o nível de abstração das representações oferecidas pela UML, a definição de quais diagramas construir, quais modelos representar, quais devem ser mantidos ou não, são definições feitas dentro do contexto de construção do software. Tais definições estão relacionadas a diversos fatores como o tamanho e disposição geográfica da equipe, a metodologia de desenvolvimento utilizada, empresa ou organização onde o software está sendo desenvolvido, e todos estes itens importam.

As abordagens e metodologias de desenvolvimento de software têm evoluído ao longo dos tempos e a modelagem (construção e uso de modelos) continua fazendo parte do processo de desenvolvimento, sendo algumas vezes a parte mais importante. Os modelos são centrais para as abordagens de desenvolvimento *model-driven*, como *Model-driven Software Engineering* (MDSE) ou *Model-driven Development* (MDD), *Agile Model-driven Development* (AMDD) e *Domain-driven Design* (DDD). Em outras abordagens mais tradicionais, os modelos são usados de diversas formas e em diversos níveis, normalmente utilizando a UML como notação principal.

MDSE, ou MDD, é uma abordagem de desenvolvimento de software onde modelos completos e extensos são criados antes da geração ou escrita de qualquer código-fonte de programa. O foco é criar e explorar modelos de domínio, que são

representações abstratas do conhecimento e atividades que governam um domínio de aplicação, ao invés de criar conceitos computacionais ou algoritmos. (Kent, 2002) (France e Rumpe, 2007) Este paradigma está normalmente relacionado ao desenvolvimento de softwares complexos, o que justifica o grande investimento na modelagem do domínio antes do investimento na solução tecnológica. (Selic, 2003) (Mellor et al. 2003) (France e Rumpe, 2007)

A abordagem DDD também coloca os modelos como peças-chave no processo de desenvolvimento de software. Baseia o *design* de softwares complexos em um modelo do domínio. Esta abordagem estimula a colaboração criativa entre especialistas técnicos e os de domínio para refinar iterativamente um modelo conceitual que aborda problemas específicos de domínio. Define um conjunto de conceitos, técnicas e princípios e seus próprios modelos e não restringe a representação de tais modelos a nenhuma notação ou forma específica. (Evans, 2004)

Alguns princípios ou guias para modelagem são apresentados em diversas publicações. (Ambler, 2002) (Pressman, 2010) (Sommerville, 2010) Tais princípios são como boas práticas ou guias em relação aos modelos de software e seu papel no processo de desenvolvimento. Temos como exemplos dessas boas práticas: “só construir modelos que sejam úteis”, “expressar claramente qual o propósito de cada modelo criado”, “não se preocupar tanto com a sintaxe do modelo. Se a comunicação do conteúdo é realizada com sucesso, a representação é secundária”. (Pressman, 2010, p.106) De qualquer forma, as decisões sobre a modelagem permanecem fortemente associadas ao contexto no qual o desenvolvimento será realizado.

No contexto desta pesquisa não consideramos nenhuma abordagem ou metodologia de desenvolvimento de software em particular, já que, primeiramente, estamos buscando identificar questões e problemas relacionados ao processo de produção e consumo de modelos no desenvolvimento de software, envolvendo especificamente o uso de (e por conseguinte a interação com) ferramentas computacionais. Consideramos neste processo os três fatores da tripla TNP. Porém, reconhecemos que a forma como a construção do software se dá, tem grande potencial para influenciar a própria maneira de realizar a modelagem de software.

2.2
Trabalhos relacionados aos fatores T, N e P

Nesta seção organizamos e discutimos alguns trabalhos realizados anteriormente, aos quais podemos relacionar os fatores da tripla TNP. Tais fatores sempre estiveram presentes no processo de modelagem de software e a sua investigação, assim como suas relações uns com os outros, de forma segmentada, nos permite focar tanto em aspectos mais pontuais (cada fator), quanto mais abrangentes (combinação de fatores).

2.2.1
Notação e Pessoas

A relação entre NOTAÇÕES e PESSOAS sempre está em pauta quando a necessidade da criação de uma representação utilizando uma notação é identificada. Através da representação, as pessoas podem externalizar e construir significados. No contexto desta pesquisa, devido ao uso de UML, um modelo está sempre associado a uma forma de representação gráfica que denota alguma porção do software. Representações gráficas sempre estiveram entre os temas favoritos de investigação de teorias cognitivas de IHC, dentre as quais a Engenharia Cognitiva (Norman, 1981) (Norman, 1986), cujo foco de descrição teórica está no usuário da representação. A Engenharia Cognitiva é uma das principais bases teóricas da abordagem mais difundida e adotada no *design* de IHC, a saber, o *Design* Centrado no Usuário.

Uma teoria cognitiva de representação gráfica é proposta por Stenning e Oberlander (1995), que define três tipos de sistemas de representação: MARS (*Minimal Abstraction Representational System*), LARS (*Limited Abstraction Representational System*) e UARS (*Unlimited Abstraction Representational System*). Estes sistemas estão relacionados à especificidade da representação, e consequentemente com sua contraparte, a abstração de tais representações. No MARS só existe um modelo cognitivo (interpretativo) para cada representação gráfica do sistema, ou seja, há uma relação 1 para 1 entre os elementos representacionais de um MARS e os elementos de domínio a que se referem. Existe abstração mínima (de fato nula) na representação. No LARS a representação denota uma classe ou categoria de elementos, ou seja, há uma relação 1 para N entre elementos representacionais e elementos de domínio. A

representação permite alguma abstração, mantendo as relações através de conceitos chave do domínio. No UARS, a representação não tem uma relação com conceitos do domínio e a abstração em relação à representação é "ilimitada". Ao contrário do MARS e do LARS, onde existe alguma relação com o domínio representado e uma chave da denotação fixa entre elementos representacionais e elementos de domínio, no UARS a chave de denotação é variante (*i.e.* a cada situação a chave pode ser alterada por algum parâmetro ou princípio específico) e é parte integrante da representação.

O LARS é cognitivamente mais eficiente para apoiar o raciocínio humano do que o MARS, pois este acomoda muito pouca ou nenhuma complexidade representacional. Também é mais eficiente do que o UARS, cuja complexidade representacional gera por si só uma carga cognitiva que se adiciona àquela que a pessoa já vai ter de despender para raciocinar sobre o domínio representado. Um LARS pode ser construído utilizando representações gráficas como, por exemplo, diagramas. Os diagramas limitam a abstração e, por isso, ajudam no processamento da representação. A informação disponível no diagrama restringe as possíveis interpretações do problema e, assim, orientam quem está raciocinando sobre o problema para tornar a solução correta. (Stenning e Oberlander, 1995) Essas características de especificidade e abstração estão relacionadas também aos modelos de software que, além de fazerem intenso uso de diagramas como forma de representação (Mnot), representam diferentes níveis de abstração (Msem) durante o processo de desenvolvimento de software.

Considerando as características de especificidade e abstração, pesquisas apresentam os modelos de software como instrumentos para raciocinar sobre problemas e suas soluções. Nos diagramas as informações são apresentadas de uma forma mais usável pelos envolvidos no raciocínio do problema, possibilitando ter diferentes pontos de vista da representação, ou diferentes focos de raciocínio sobre a representação. (Funt, 1995) Nos diagramas também temos a representação das relações entre os elementos, sendo essas relações cruciais para o raciocínio sobre o problema. O agrupamento de informações relacionadas também é uma característica dos diagramas, o que ajuda nas inferências para solução de problemas, inferências perceptivas que são fáceis para os seres humanos. (Larkin e Simon, 1995) Nem sempre os diagramas são benéficos, especialmente se são mal construídos ou mal usados. É por esta razão que o estudo de diagramas deve

ser aprofundado como uma contribuição teórica para questões práticas enfrentadas por *designer*s. (Blackwell, 2001)

Com foco na notação definida para o sistema de representação, destacamos duas pesquisas com perspectivas e objetivos bastante distintos. As "*Cognitive Dimensions of Notations*" (CDN) foram propostas por Thomas Green e Alan Blackwell (1998) como uma ferramenta para *design* e avaliação de artefatos de informação, apresentando um vocabulário que possibilita a discussão das questões cognitivas envolvidas na construção de significados, particularmente em artefatos de software. (Blackwell e Green, 2003) As CDN podem ser utilizadas em qualquer sistema de representação, seja ele somente visual ou combinado com outras representações, como a textual. Outra proposta de ferramental para *design* e avaliação de sistemas de representação é chamada de *The Physics" of Notation.* (Moody, 2009) É um conjunto de princípios, sintetizados da teoria e de conhecimentos empíricos, focados em propriedade físicas da notação ao invés de propriedades semânticas. Podem ser usadas para avaliar, comparar e melhorar notações visuais assim como para construir novas notações. Ela, porém, só considera a representação visual e tem foco maior na sintaxe da notação, provendo diretrizes prescritivas de *design* e previsão de métricas de avaliação de sistemas notacionais.

Petre (1995) apresenta reflexões interessantes sobre a combinação de diferentes notações para representar artefatos no processo de construção de software. Ela argumenta que existem características positivas e negativas de uma representação basicamente visual e discute que não basta ver a representação para ser capaz de construir algum entendimento, algum significado a partir dela. Com foco na construção de significados, ela aponta algumas características cognitivas que uma representação gráfica precisa ter para ser interpretada de forma confiável, para que todos os envolvidos tenham uma intepretação convergente e, portanto, para que a representação comunique a todos a mesma mensagem. Ela argumenta o poder da combinação de representações (gráfica e textual) e comenta sobre a necessidade muitas vezes comum de estender a notação usada para compor representações mais completas, que remetam melhor àquilo que é o seu significado.

Em outra publicação Petre (2013a) aponta a necessidade de instrumentos que sejam capazes de apoiar a transformação ou evolução dos modelos de

software de um nível de abstração para outro (ex. casos de uso para classes) e de um tipo de audiência para outro (ex. contexto do negócio para contexto da tecnologia). Sendo tais modelos usados como instrumentos de comunicação, eles precisam de elementos que façam a ponte de um ponto a outro. Essa ponte lida com questões da notação (Mnot) e da semântica (Msem) do modelo de software. A área de *Computer Supported Cooperative Work* (CSCW) apresenta o conceito de *boundary objects* (objetos limítrofes) (Star, 1990) que são entidades que conectam comunidades permitindo que diferentes grupos colaborem em uma tarefa em comum. Os objetos limítrofes permitem o entendimento e negociação de entre pessoas de diferentes comunidades envolvidas em uma tarefa colaborativa. São elementos chave no compartilhamento e construção de conhecimento colaborativo (Arias e Fisher, 2000) (Lee, 2005), apresentando características como a de flexibilidade interpretativa associada a como e quando um objeto limítrofe é usado. Além disso, possibilita que pessoas trabalhem juntas sem a necessidade de um consenso, mas sim de um compartilhamento de entendimentos. (Star, 2010) Os modelos de software têm essa característica de representar um significado coletivo e serem usados como instrumentos de comunicação entre diferentes comunidades (Msem). Portanto, funcionam como um tipo ou uma coleção de objetos limítrofes (Mnot) do processo de desenvolvimento de software.

2.2.2
Ferramenta e Notação

Na relação entre FERRAMENTA e NOTAÇÃO, destacamos trabalhos de MDSE, que é a área que mais apresenta pesquisas sobre modelos de software. Porém, o foco dessas pesquisas está predominantemente na notação utilizada para construção de modelos, a UML, e a transformação de modelos de tal forma que seja possível a geração automática de código-fonte a partir deles. Dentro da metodologia MDSE, as pesquisas estão concentradas na etapa de automação e realização. (Figura 4 adaptada de Brambilla et al., 2012) Até onde nossa pesquisa verificou a etapa de modelagem em si, onde os modelos são construídos para representar o domínio do problema relacionado ao software, não é tema presente nas pesquisas de MDSE. Sobre a transformação de modelos mais abstratos em modelos que possibilitem a geração do código, há diversos trabalhos que propõem soluções como uma sistemática baseada em semântica para transformação de

modelos (Lano e Kolahdouz-Rahimi, 2011), um modelo genérico de transformação de modelos, considerando o reuso (Cuadrado et al., 2011) e a remoção automática de redundância em diagrama de classes (Maraee e Balaban, 2014). Porém, esses trabalhos partem do princípio de que o modelo a ser transformado, aquele que representa o domínio do problema relacionado ao software em desenvolvimento, é um 'bom' modelo, que representa o que deve ser tratado pelo software de forma completa e correta em relação do domínio. Os diferentes usos dos modelos ao longo do desenvolvimento de software são destacados e reconhecidos, porém o foco da discussão é a possibilidade de transformação dos modelos em código. (Zheng e Taylor, 2013) Como este 'bom' modelo deve ser construído ou como deve ser a ferramenta de apoio para a construção deste modelo, não são temas abordados. Quando se fala em ferramentas, o foco fica em torno da etapa de Automação para a Realização da metodologia MDSE. (Bruneliere e Cabot, 2014) A transformação de modelos que podem gerar código presume que o modelo original representa o que precisa ser transformado em código. (Kühne, 2006) (Muller et al., 2012)

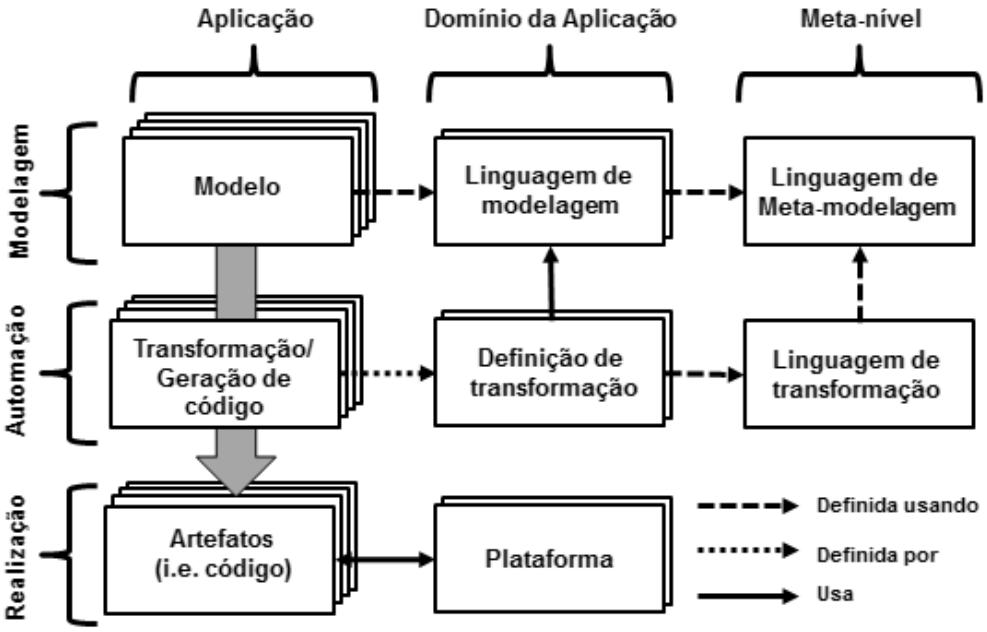


Figura 4. Metodologia MDSE

A premissa de ter um 'bom' modelo vem sendo questionada pela própria comunidade de MDSE. Whittle et al. (2013) já apontam problemas relacionados ao processo de construção de modelos, indicando que outros fatores devem ser considerados, além das ferramentas, para que o modelo represente o domínio

desejado, levando em conta a notação e a semântica deste modelo. Outro questionamento pode ser encontrado no discurso do *designer* de uma das ferramentas de modelagem de software avaliada nesta pesquisa (Astah). Kenji Hiranabe concorda que uma abordagem orientada a modelos é benéfica para o desenvolvimento de software, porém destaca que os modelos para comunicação humano-computador não são os principais modelos da abordagem *model-driven* e sim aqueles usados para a comunicação humano-humano. (Figura 5)

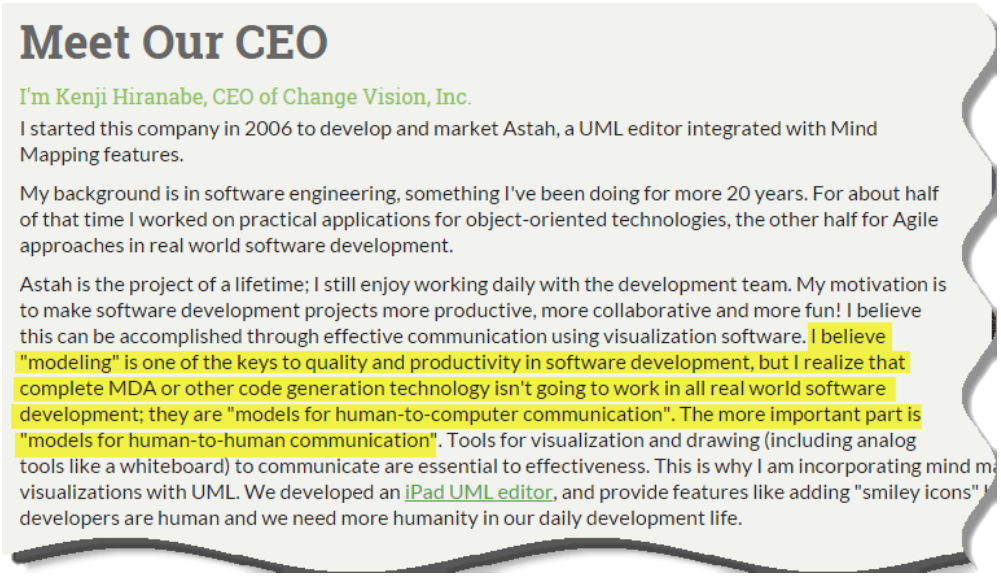


Figura 5. Message from Astah's *designer*

Mussbacher et al.(2014) ressalta a importância do processo de construção do modelo (Msem), falando sobre "*modeling for the Illiterate*" (Models4 Δ○ᗡ;) que trata de linguagens e notações (Mnot) de modelagem para pessoas que não sabem construir modelos (Msem). Discute como ensinar pessoas a usar a notação para construir significados através de modelos. A ideia de promover o entendimento sobre o que é realmente a modelagem é abordada nesta pesquisa. Esta também destaca que a comunidade MDE (*Model-Driven Engineering*) tem focado seus esforços nas ferramentas e nos seus próprios modelos de software (Mnot), sem considerar as atividades de modelagem de outras áreas, como, por exemplo, a área de IHC. Neste caso, já temos a indicação de PESSOAS relacionadas aos demais fatores, ponto que será discutido na próxima seção.

2.2.3
Ferramenta, Notação e Pessoas

A relação entre os fatores FERRAMENTA, NOTAÇÃO e PESSOAS identificada nos trabalhos relacionados se dá de duas formas: (1) PESSOAS é um fator fortemente presente nas pesquisas apresentadas; e (2) o fator PESSOAS pode ser inferido pela pesquisa apresentada ou é um fator identificado, porém não tratado especificamente na pesquisa. Mesmo com graus de relevância diferentes, o fator PESSOAS foi identificado e discutido em relação aos dois outros fatores da TNP.

O processo de *design* de software apresenta conceitos e perspectivas interessantes que podem ser considerados de forma análoga para a produção de modelos de software, considerando também o suporte computacional para produção de tais modelos. O processo de *design* naturalmente relaciona os três fatores TNP. O modelo é um artefato intelectual, construído por uma ou mais pessoas. Esse processo de construção apresenta questões relacionadas à criatividade e ao *design rationale*. (Carroll, 2012) O suporte computacional para a construção de um artefato intelectual deve contar com algumas características, como viabilizar a constante evolução ou mudança do artefato, a possibilidade de brincar ou experimentar com as representações do artefato, e manter o *rationale* das mudanças relevantes feitas no artefato. (Carroll, 2012) O processo de *design rationale*, que se beneficia muito de um suporte computacional, pode gerar novas soluções de *design* promovendo a conversa e troca de ideias entre os envolvidos. Portanto, os eventos que ocorrem durante o processo de *design* devem ser mantidos e relacionados ao artefato. Assim, as tomadas de decisão e experimentações de *design* do artefato podem ser realizadas pelos envolvidos quando necessário. (McCall, 2013) Para oferecer suporte à criatividade durante o processo de *design*, deve ser possível a comparação entre artefatos diferentes (cenários, modelos, protótipos) para avaliar os méritos e limitações de cada um deles, se podem ser combinados ou se podem se complementar, criando assim um artefato com a representação desejada pelos responsáveis pelo *design*. (Sutcliffe, 2013)

Como já foi dito, a UML é o objeto central de investigação de diversas pesquisas sobre o uso da notação no contexto de desenvolvimento de software. Nesses casos, temos os três fatores TNP em jogo. Seu uso foi investigado em diferentes momentos de sua história e com diferentes versões, como é o caso de UML 1.5 (Dobing e Parsons, 2006) e UML 2.0 (Petre, 2013). Porém, alguns

resultados, mesmo passados anos entre uma pesquisa e outra, ainda permanecem os mesmos como, por exemplo, a complexidade desnecessária da notação UML, alguns diagramas da UML poderem ser usados para comunicar com pessoas fora do contexto técnico (*stakeholders*) e ser feito um uso parcial dos diagramas UML. As duas pesquisas citadas também reportam sobre ferramentas usadas para a modelagem com a UML, sendo que a de 2006 indica a necessidade de haver suporte computacional, enquanto a de 2013 apresenta relatos de profissionais cuja maioria indica que não consideram as ferramentas de modelagem parte importante do desenvolvimento de software.

A investigação do uso da UML na prática do processo de desenvolvimento de software é uma necessidade reportada tanto por pesquisadores, quanto por profissionais. (Petre, 2013b, 2014) Somente em contexto de uso será possível verificar como esta notação pode ser adaptada, como as pessoas se apropriam, ou não, dela durante o uso, como o suporte computacional impacta o uso da notação, como as pessoas criam representações por meio e a partir dela e outras observações que só podem ser feitas considerando os três fatores TNP simultaneamente e em relação uns com os outros. Apesar da indicação de que existem vários problemas relacionados aos modelos UML, pesquisadores tendem a considerar a UML como dado da questão e parecem relutantes em fazer perguntas que podem tornar a UML mais eficaz. (Budgen, 2011)

Petre e van der Hoek (2013) reuniram várias pesquisas realizadas sobre dados coletados durante um workshop realizado com várias equipes de *design*, para as quais foi proposta a tarefa de pensar numa solução de *design* para um determinado problema (criar um simulador de sinal de trânsito). O objetivo desse workshop era o de observar técnicas, práticas e métodos utilizados pelas equipes para discutir e tomar decisões sobre o *design* de uma solução de software. O foco de investigação era compreender o *design* de software feito logo no início do processo de desenvolvimento (*early design*), o qual apresenta algumas características particulares: (1) o centro das atenções tende a ser sobre decisões importantes que definem a aplicação de software como um todo, definindo que funcionalidades devem ser oferecidas e como; e (2) tais decisões são tomadas através de conversas, usando-se diagramas auxiliares, *sketches*, e outros artefatos para apoiar essas conversas.

Algumas perguntas guiaram as pesquisas ao longo desse workshop (Petre e van der Hoek, 2013): Que representações *designer*s de software usam nesse estágio inicial do *design*? Quais são as estratégias utilizadas para navegar pelo vasto espaço de possibilidades de *designer* disponíveis? Como os *designer*s de software colaboram e se comunicam para resolver um problema de forma eficaz? Como eles acomodam o contexto na sua prática e raciocínio? Quais são os desafios particulares relacionados ao *design* inicial em comparação com o projeto conceitual de outras disciplinas?

A dinâmica do workshop contou com o apoio de um quadro branco para que os *designer*s pudessem registrar e representar o que fosse definido em relação ao *design* da solução. Mesmo sem a utilização de um suporte computacional, a relação do contexto e da atividade realizada foi levada em consideração para a discussão sobre o uso de uma ferramenta para registro das decisões de *design*. Quando a equipe de *design* se encontra concentrada, compartilhando o mesmo contexto e espaço físico, o suporte computacional pode ser opcional, tendo como possibilidade reunir todos numa sala e utilizar outros recursos como o próprio quadro branco, notas afixadas na parede e etc. Porém, quando isso não é possível, o suporte computacional para apoiar esse processo de *design* se torna indispensável. (Budgen, 2013)

As representações utilizadas pelos *designer*s também foram objeto de investigação desse workshop. A UML foi uma nas notações utilizadas, porém a estratégia de combinar notações (UML, ER, prototipação, etc.) com desenhos e anotações foi destacada como uma prática que busca apoiar o processo de comunicação entre os envolvidos na discussão para definir o *design* do software. A notação utilizada buscava prover fluidez à comunicação e ser flexível para que mudanças fossem feitas rapidamente, sem preocupação com formalismos para não atrapalhar o processo de *design* (Petre, 2013a)

Esta coletânea de estudos (Petre e van der Hoek, 2013), realizados sobre o mesmo conjunto de dados coletados durante a dinâmica das equipes de *design* (vídeos e transcrições), mostra como a prática de *design* de software é uma atividade rica para investigação por diferentes ângulos e considerando diversos fatores. Com o mesmo conjunto de dados, 22 pesquisadores ou grupos de pesquisadores realizaram investigações e observações sobre a prática de *design* em ação, levando em conta tudo o que pode ter influência nessa prática, todo o

contexto de execução da atividade de *design*. Os fatores FERRAMENTA, NOTAÇÃO e PESSOAS foram identificados e discutidos, assim como as suas relações dentro do processo de *design*.

Considerando que o desenvolvimento de software é um contexto altamente colaborativo, pesquisas na área de CSCW (*Computer-Supported Cooperative Work*) apresentam discussões e características das etapas do processo de desenvolvimento onde o suporte a colaboração é fundamental. (de Souza et al., 2005) (de Souza et al., 2007) A modelagem de software é apresentada como uma atividade inerentemente colaborativa, onde vários *designer*s de software colaboram para a confecção de um mesmo artefato ou para elaboração de artefatos relacionados. De qualquer forma, os envolvidos devem estar cientes das atividades dos demais durante a produção dos artefatos. O registro do *rationale* das decisões tomadas ao longo do desenvolvimento de software, tanto decisões mais técnicas (ex. definição de arquitetura, linguagem) quando do registro de decisões de *design* (*design rationale*) também são destacados e discutidos como de grande importância e que necessitam de suporte computacional, considerando seu contexto colaborativo. (Whitehead, 2007) A colaboração para construção de artefatos de software cria uma relação de dependência sócio-técnica caracterizada pela relação entre artefatos de software e *designer*s que precisam utilizar, algumas vezes, os mesmos artefatos. Ter consciência desta relação é crucial para que a colaboração no desenvolvimento de software seja mantida, sem conflitos e necessidade de retrabalho. (de Souza et al., 2007) A necessidade da criação de um protocolo social para compensar ou complementar o protocolo técnico, também é um ponto relacionado à característica colaborativa do desenvolvimento de software, e está relacionado aos fatores T, N e P da tripla. (Ellis, Gibbs, 1991)

O fator PESSOAS tem sido citado em recentes pesquisas de MDSE como de grande importância para a adoção da metodologia *model-driven*. A modelagem de software deve considerar alguns fatores que não podem ser isolados. Eles que dão conta do artefato em si e do contexto em que o artefato é construído. (Whittle et al., 2013) (Whittle et al., 2014) Em relação as ferramentas de modelagem, destacam a necessidade de adaptação de ferramentas às pessoas e não das pessoas às ferramentas. Para isso, *designer*s de ferramentas que apoiam o MDSE devem buscar conhecimentos da área de IHC para desenvolver ferramentas que atendam

às necessidades das pessoas envolvidas no processo de desenvolvimento. (Whittle et al., 2013)

2.3
Sobre este capítulo

Neste capítulo apresentamos os trabalhos relacionados que tomamos como referência para o desenvolvimento desta pesquisa. Através da organização destes trabalhos considerando os fatores FERRAMENTAS, NOTAÇÃO e PESSOAS, buscamos posicionar a nossa pesquisa no cenário corrente das áreas e comunidades que nos indicaram caminhos para esta investigação e que se beneficiariam com respostas à questão desta pesquisa: *Como ferramentas e notações de modelagem influenciam e apoiam o processo intelectual de produção e consumo de modelos de software através da interação humano-computador*?

3
Ferramental Metodológico

Este capítulo apresenta e detalha o ferramental metodológico produzido e usado nesta pesquisa para investigar e caracterizar a problemas de interação de ferramentas de modelagem durante a produção e consumo de modelos de software. Avalia como a interação com ferramenta de modelagens se dá (ou seja, a comunicabilidade e usabilidade das interfaces) e tenta antecipar como problemas de comunicabilidade e usabilidade podem afetar a qualidade dos modelos e a função que eles desempenham no processo de desenvolvimento de software. Este ferramental é composto por um método que combina as perspectivas semiótica e cognitivas de IHC para a avaliação de ferramentas, o *Combined Semiotic-Cognitive Evaluation method* (CSCE), e a tripla *tool-notation-people* (TNP), uma estrutura única que articula uma abrangente caracterização dos fenômenos que ocorrem durante o processo de desenvolvimento de software, particularmente relacionados à modelagem. Ela considera os três fatores usados em sua denominação - Ferramenta (T), Notação (N) e Pessoas (P) – além das relações entre eles.

O CSCE, associado à TNP, foi utilizado para avaliar ferramentas de modelagem de software selecionadas para os estudos nesta tese. A tripla TNP também foi utilizada como referência para organizar e discutir trabalhos relacionados a esta pesquisa, conforme apresentado no capítulo anterior. Além disto, foi usada como recurso de articulação para análise dos dados coletados nos estudos realizados. Seus fatores foram considerados como partes de um todo e que não podem ser observados isoladamente para o contexto de modelagem de software.

Primeiramente, apresentamos alguns conceitos da Engenharia Semiótica necessários para a contextualização da estratégia escolhida para a investigação nesta pesquisa, seguida da apresentação separada dos dois componentes do ferramental metodológico: a tripla TNP e o CSCE, respectivamente. Ao final,

apresentamos de forma detalhada e completa o método que associa a TNP e o CSCE, construído para proceder à investigação apresentada nesta tese.

3.1
Engenharia Semiótica como base

A Engenharia Semiótica (de Souza, 2005), uma das teorias da área de IHC propostas para caracterizar o fenômeno de interação humano-computador, é a base conceitual desta pesquisa. Diferente da maior parte das teorias de IHC que adotam perspectivas cognitivas, a Engenharia Semiótica, assim como seu nome indica, adota uma perspectiva semiótica. Ela considera a semiótica conforme definida por Eco como sendo a disciplina que estuda a significação e a comunicação. (Eco 1976, 1983) Alguns **conceitos da semiótica** precisam ser apresentados, antes da própria Engenharia Semiótica. São eles: **signo, significação e comunicação**.

Signo é tudo aquilo que, sob certo aspecto ou em certa medida, representa algo para alguém (Peirce, 1992-1998). S**ignificação** é o processo de formação de signos por associações sistemáticas entre expressões e conteúdos (que assim estabelecem 'sistemas de significação'), resultantes de convenções sociais e culturais adotadas por intérpretes e produtores destes signos. **Comunicação** é o processo através do qual, para uma variedade de propósitos e efeitos, produtores de signos expressam os significados pretendidos, através da exploração das possibilidades de sistemas de significação existentes. Ocasionalmente, eles podem até recorrer a signos não convencionais, por exemplo, ao inventar um novo signo ou ainda usar um signo conhecido de modo imprevisto para realizar certo efeito retórico e alcançar sua intenção de comunicação de maneira mais eficaz ou eficiente. Sempre que a cultura produz uma convenção que relaciona um conjunto expressivo de elementos a um conjunto de conteúdos, um novo sistema de significação é constituído e um novo código torna-se disponível para comunicação (Eco, 1976).

A semiótica segundo Eco, também define três elementos fundamentais na comunicação: **intenção** (o que se quer alcançar com a comunicação), **conteúdo** (que informações são usadas) e **expressão** (quais formas e meios de comunicação são escolhidos). Com esses conceitos da semiótica apresentados, podemos, então, passar à apresentação da Engenharia Semiótica.

A Engenharia Semiótica (de Souza, 2005) considera a interação humano-computador como um processo de comunicação entre humanos mediada por computador, onde temos três partes envolvidas: *designer*, usuário e software. A comunicação acontece entre *designer* e usuário através do software em tempo de interação. O software é o preposto do *designer* (Figura 6 de PIMENTEL et al., 2011) e efetua somente os caminhos de conversa que foram antecipados e projetados pelo *designer* durante o desenvolvimento do software. O software representa e age em nome do *designer* neste processo mediado de comunicação.

Figura 6. O preposto do *designer*

O termo ***designer*** está sendo usado, de fato, para representar os 'criadores' de artefatos computacionais interativos em sentido coletivo. Ocasionalmente pode se tratar de uma única pessoa, mas é comum que se trate de uma equipe de profissionais. Nestes casos, o *designer*, singular, é aquele que desempenha o papel de porta-voz de toda a equipe. O mesmo acontece no caso do termo **usuário**, já que a possibilidade de haver diversos usuários (*i.e.* um conjunto de perfis distintos de usuários) é bastante comum. Sendo assim, os termos *designer* e usuário representam o coletivo de tais papéis envolvidos no processo de comunicação.

Durante a interação do usuário com o software, existe um processo de metacomunicação, que é a comunicação sobre como, quando, onde e por que se comunicar com o sistema idealizado pelo *designer*. Esse tipo específico de

metacomunicação é definido como uma mensagem unidirecional do *designer* para o usuário, que apresenta informação sobre quem o *designer* acredita que é o usuário, suas necessidades, preferências e objetivos, e também quais são as decisões do *designer* para atender esse usuário visado.

Essa mensagem do *designer* para o usuário é representada pelo template de metacomunicação proposto pela Engenharia Semiótica:

"Eis a minha visão de quem é você, que aprendi que você deseja ou precisa fazer, de que formas preferenciais e por quê. Este é o sistema que consequentemente elaborei para você, e esta é a forma como você pode ou deve usá-lo para realizar um conjunto de objetivos que se enquadram nesta visão."

O *template* de metacomunicação é uma forma abstrata de representar o conteúdo completo da mensagem que os *designers* comunicam através da interface para os usuários, à medida que os usuários interagem com o software. Através do preenchimento do *template*, temos uma representação explícita do entendimento do *designer* em relação ao usuário, suas necessidades, preferências e objetivos e, associadas a isso, as decisões tomadas pelo *designer* na concepção do software para atender aos anseios desse usuário concebido pelo *designer*. Com o uso do *template*, temos uma forma de representar, de acordo com a expressão palpável da visão do *designer* (através de signos da interface do software), o usuário visado, os conteúdos com que este usuário precisa interagir para controlar o software e realizar várias tarefas, a forma como esse usuário deseja ou prefere se expressar, tendo em vista as suas intenções.

O modelo do espaço de comunicação adotado pela Engenharia Semiótica para estruturar o espaço de *design* de IHC foi o proposto por Jakobson (1960) para caracterizar funções da comunicação social humana em geral (Figura 7 - Barbosa e Silva, 2010). A comunicação é orientada por uma **intenção** do emissor, os efeitos que o emissor quer provocar ao transmitir o **conteúdo** da sua mensagem ao receptor. Para que a comunicação seja bem sucedida, o emissor deve selecionar cuidadosamente uma **expressão** para o conteúdo que deseja comunicar, utilizando um código que o receptor seja capar de interpretar e, no caso de IHC, que o sistema computacional seja capaz de processar. (Barbosa e Silva, 2010)

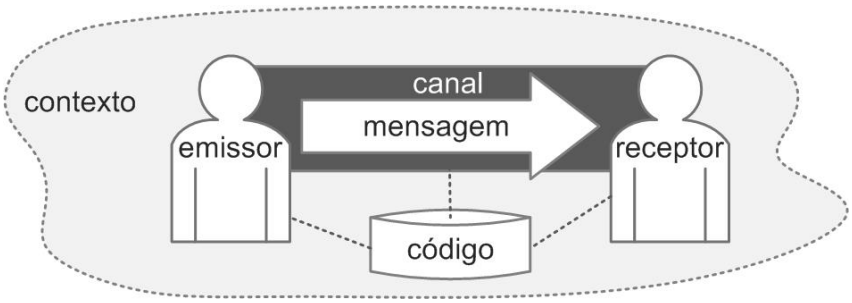


Figura 7. Modelo de espaço de comunicação de Jakobson

Considerando esses elementos do espaço de *design*, o *designer* precisa examinar todas as suas partes e tomar decisões sobre cada elemento, respondendo as seguintes perguntas (de Souza, 2005, p.87):

- **Quem é o emissor (*designer*)?** Que aspectos das limitações, motivações, crenças e preferências do *designer* devem ser comunicados ao usuário para o benefício da metacomunicação;
- **Quem é o receptor (usuário)?** Que aspectos das limitações, motivações, crenças e preferências do usuário, tal como interpretado pelo *designer*, devem ser comunicados aos usuários reais para que eles assumam seu papel como interlocutores do sistema;
- **Qual é o contexto da comunicação?** Que elementos do contexto de interação — psicológico, sociocultural, tecnológico, físico etc. — devem ser processados pelo sistema, e como;
- **Qual é o código da comunicação?** Que códigos computáveis podem ou devem ser utilizados para apoiar a metacomunicação eficiente, ou seja, qual deve ser a linguagem de interface;
- **Qual é o canal?** Quais canais de comunicação estão disponíveis para a metacomunicação *designer*–usuário, e como eles podem ou devem ser utilizados;
- **Qual é a mensagem?** O que o *designer* quer contar aos usuários, e com que efeito, ou seja, qual é a intenção comunicativa do *designer*.

A mensagem é construída pelo emissor utilizando o código de comunicação estabelecido, e faz parte deste código a sua própria gramática. A forma como os

elementos do código de comunicação são combinados pelo *designer*, é uma parte crucial da mensagem para que a comunicação seja bem sucedida. A menos que o usuário (*i.e.* receptor) consiga aprender esta gramática e 'falar' a linguagem de interface que é, em última análise, este código, ele simplesmente não consegue conversar com o preposto do *designer* (*i.e.* não consegue realizar nenhuma intenção própria, comunicar nenhum conteúdo próprio, simplesmente porque não consegue dominar as regras de expressão da interface). (de Souza, 2005, p.10) Com um domínio inadequado desse sistema de expressão definido artificialmente e convencionado por uma tecnologia (*i.e.* a linguagem da interface), o usuário pode não ser capaz de se beneficiar de tal tecnologia que, idealmente, foi desenvolvida para resolver algum problema ou facilitar atividades deste usuário. Sendo a tecnologia um software, o *designer* vai emitir a mensagem ao usuário através da interface deste. Durante a interação, o usuário vai receber essa mensagem e interpretá-la. O ideal é que entre a emissão (pelo *designer* através da interface) e a recepção (do usuário durante interação com a interface) a mensagem permaneça a mesma, que não haja ruídos de comunicação provocados por problemas de comunicabilidade da mensagem. O sucesso dessa comunicação está relacionado ao receptor da mensagem (usuário) conseguir aprender a gramática da interface, definida pelo *designer* no momento de construção do software, e ser capaz de se comunicar através dessa interface. A comunicação *designer*-usuário através da interface está relacionada ao conceito de comunicabilidade da Engenharia Semiótica, que será apresentado em detalhes mais adiante neste capítulo.

A Engenharia Semiótica investiga processos de comunicação em dois níveis distintos (Figura 8 adaptada de Barbosa e Silva, 2010): a comunicação direta usuário–sistema e a metacomunicação do *designer* para o usuário mediada pelo software, através da sua interface. No nível da comunicação direta usuário–sistema (Figura 8-1) investigamos a recepção da mensagem, os usuários percebem a mensagem do *designer* na medida em que interagem com o sistema. No nível da metacomunicação do *designer* para o usuário mediada pelo software, através da sua interface (Figura 8-2), investigamos a emissão da mensagem do *designer* para o usuário. O *designer* conta aos usuários através da interface a sua visão sobre quem eles são, do que precisam, o que desejam, como preferem interagir e para que usam o software.

Figura 8. Processo de comunicação na Engenharia Semiótica

A comunicabilidade, conceito chave da Engenharia Semiótica, pode ser tecnicamente definida como a capacidade do preposto do *designer* (interface de software) para atingir a metacomunicação completa, transmitindo aos usuários a essência da mensagem original do *designer*. (de Souza, 2005, p. 114) Esta metacomunicação será eficiente e eficaz se a comunicação é organizada e se oferece múltiplos recursos para serem escolhidos conforme a situação (eficiente) e se alcança o resultado desejado (eficaz). (de Souza et al., 2010) A comunicabilidade está relacionada à responsabilidade do *designer* em comunicar ao usuário suas intenções de *design* e a lógica que rege o comportamento da interface. Esse conceito se pauta no pressuposto de que, se o usuário tiver acesso à lógica de *design*, ele terá melhor condição de fazer um uso produtivo e criativo do apoio computacional oferecido pelo software. (Barbosa e Silva, 2010, p. 28) Como discutido anteriormente neste capítulo, essa lógica de *design* está relacionada com a gramática da linguagem da interface definida pelo *designer*. Esta gramática também compõe a mensagem do *designer* para o usuário através de seu preposto (*i.e.* interface do software) e influencia a comunicabilidade deste preposto.

O conceito de comunicabilidade foi alterado e complementado conforme a Engenharia Semiótica se estabelecia como teoria, com seus conceitos e métodos sendo utilizados em pesquisas e práticas de IHC. Definida originalmente em 2000 como "a qualidade distintiva de sistemas computacionais interativos que comunicam eficiente e eficazmente ao usuário suas intenções de *design* e princípios interativos" (Prates et al. 2000, p. 32), a comunicabilidade se referia principalmente às condições operacionais. Naquele momento, se buscava

principalmente, detectar evidências da comunicação de intenções de *design* e princípios de *design* em artefatos computacionais interativos. Entretanto, os interlocutores envolvidos no processo não estavam explícitos nesta definição.

Em 2005, foi discutida a formulação metonímica do conceito de comunicabilidade definido no parágrafo anterior. A definição transferiu para o *design* (produto) a capacidade que de fato é esperada do *designer* (produtor) (de Souza, 2005, p. 113). Na discussão da metonímia incorporada na definição anterior, foi proposta uma nova definição: Comunicabilidade é a capacidade do preposto do *designer* para alcançar a metacomunicação completa, transmitindo aos usuários a essência da mensagem original do *designer*. Ela se aplica aos códigos interpretativos e expressivos que o preposto do *designer* manipula para geração e interpretação de mensagens durante a interação com usuários. (de Souza, 2005, p. 114) Esta definição conecta a comunicabilidade mais claramente na ontologia da Engenharia Semiótica. Ao enfatizar que o processo de comunicação em questão é entre usuário e preposto de *designer*, é reiterada a noção que, nesta teoria, *designer* e usuário compartilham o mesmo estado ontológico – eles são interlocutores em tempo de interação.

A redefinição mais recente do conceito de comunicabilidade foi feita buscando esclarecer mal-entendidos sobre a comunicabilidade e os métodos utilizados para avaliá-la, em relação aos significados de eficiência e eficácia da metacomunicação. Para isso, foi incluída a definição de comunicação eficiente e eficaz, no contexto da comunicabilidade como sendo: uma comunicação organizada e que oferece múltiplos recursos (eficiente), capaz de alcançar o resultado esperado (eficaz) (de Souza et al., 2010).

Depois desta nova definição, o conceito de comunicabilidade, atualmente considerado, é: a capacidade do preposto do *designer* (interface de software) em atingir a metacomunicação completa, transmitindo aos usuários a essência da mensagem original do *designer*.

3.2
Detalhamento do Ferramental Metodológico

Nesta pesquisa começamos a explorar o conceito de comunicabilidade de modelos de software, investigando a produção e consumo desses modelos com o apoio de ferramentas de modelagem de software. Entendemos que a investigação

e exploração deste conceito poderiam ser feitas num contexto mais amplo, com ou sem o apoio computacional. Porém, nesta pesquisa realizada na área de IHC, estamos focando na modelagem realizada com apoio de ferramentas, onde buscamos investigar como problemas de IHC destas, identificados durante a produção e consumo de modelos, podem impactar na comunicabilidade dos artefatos em si.

O contexto desta pesquisa é o processo de produção e consumo de modelos no desenvolvimento de software. Os desenvolvedores constroem modelos de software utilizando ferramentas computacionais de modelagem. Tais ferramentas são, por sua vez, desenvolvidas por outros desenvolvedores de software. Portanto, os desenvolvedores, atuando como usuários de ferramentas de modelagem de software, estão expostos a questões de IHC que qualquer usuário de um sistema computacional está. Porém, como o seu objetivo final com a construção dos modelos é a construção de um software, essas questões de IHC podem potencialmente impactar os artefatos que orientam o desenvolvimento do software final e, possivelmente, o software final. Sendo assim, consideramos as seguintes premissas:

1) É possível que, se os desenvolvedores de software tiverem problemas de IHC com suas próprias ferramentas, então a qualidade do seu produto pode ser impactada, e;
2) É possível que, se a qualidade do produto for impactada, então o usuário final experimente problemas de interação que estejam direta ou indiretamente ligados aos problemas de IHC experimentados pelos desenvolvedores.

Com esta pesquisa buscamos identificar evidências e indícios sobre as premissas (1) e (2). Somente após conhecer e identificar evidências sobre tais premissas será possível investigar quais dos problemas de IHC do usuário estão ligados a problemas de IHC dos desenvolvedores, caracterizando a propagação de problemas de interação durante o processo de desenvolvimento para a interação do usuário final com o software desenvolvido. A relação de problemas de IHC enfrentados por desenvolvedores enquanto constroem o software com problemas de IHC enfrentados pelo usuário final deste software, caracterizando uma

propagação de problemas de IHC, é a questão em longo prazo que motivou a pesquisa desta tese.

Neste cenário de propagação de problemas de IHC temos dois momentos de interação: durante o processo de desenvolvimento, quando desenvolvedores de software são usuário de ferramentas de modelagem, e durante a interação do usuário final com o software construído pelo desenvolvedor, usuário de ferramentas de modelagem (Figura 9). Buscando evidências e indícios para a premissa (1) descrita acima, procuramos identificar, caracterizar e discutir problemas de interação que ocorrem durante o processo de desenvolvimento, considerando modelos de software e as ferramentas utilizadas para sua construção e uso. Este espaço de investigação está representado pela parte inicial da Figura 9, onde o desenvolvedor de software é o usuário de ferramentas de modelagem.

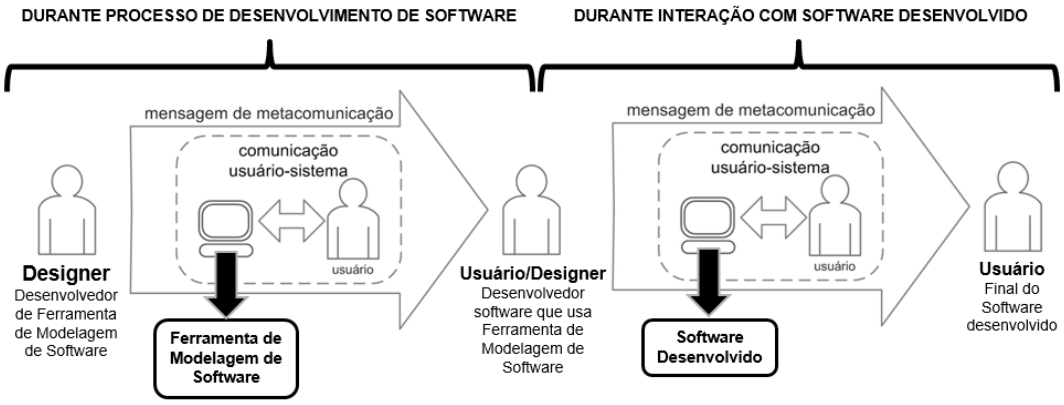


Figura 9. Desenvolvedor de software também é usuário

Nosso objetivo com esta pesquisa é obter um conhecimento mais profundo, contextualizado e evidenciado da experiência do desenvolvedor de software como usuário de ferramentas de modelagem de software, considerando as características do processo de desenvolvimento de software como uma atividade colaborativa, onde diversos tipos e perfis de pessoas têm um objetivo comum que é o software em desenvolvimento. Com este conhecimento buscamos evidências e indícios para a premissa (1) e apontar caminhos para a investigação da premissa (2). Para alcançar este objetivo, desenvolvemos a **tripla *tool-notation-people* (TNP)** e o ***Combined Semiotic-Cognitive Evaluation method* (CSCE)** que apresentados em detalhes a seguir.

O desenvolvimento da tripla TNP e do CSCE viabilizou a realização desta pesquisa para identificar evidências e indícios de (1) e encaminhar (2). E também se apresentou como um ferramental metodológico poderoso para, futuramente, ser usado na investigação da questão em longo prazo sobre a propagação de problemas de IHC, que é a relação de problemas de IHC enfrentados por desenvolvedores, enquanto constroem o software, com problemas de IHC enfrentados pelo usuário final deste software.

3.2.1
Tripla Tool-Notation-People (TNP)

A tripla *tool-notation-people* (TNP) é um recurso de articulação de três fatores fortemente relacionados no contexto de desenvolvimento de software: Ferramentas (T), Notação (N) e Pessoas (P). Além dos fatores em si, as relações entre cada um deles e entre todos, combinados, também são consideradas na caracterização do contexto de desenvolvimento de software em que se produzem e consomem modelos representativos de diferentes aspectos do artefato final que será entregue aos usuários.

Algumas características particulares deste contexto nos motivaram a desenvolver a tripla TNP, relacionando FERRAMENTA, NOTAÇÃO E PESSOAS, como recurso de articulação. Tais características são:

Suporte computacional é comum. Em várias etapas do processo de desenvolvimento de um software, são utilizadas ferramentas computacionais para construir os artefatos necessários ao desenvolvimento. No caso de modelos de software, em algumas etapas mais precoces do processo, em que o domínio onde o novo software será inserido ainda está sendo compreendido, o uso destas ferramentas pode ser dispensado, dando-se preferência a outros meios de representação mais ágeis como quadros brancos ou *smartboards*, papel, e outros recursos. Isto facilita que os envolvidos conversem e busquem um entendimento sobre o software a ser desenvolvido. (Petre e Van der Hoek, 2013) Porém, já nesta fase, a decisão pode depender de diversos fatores como complexidade do projeto de desenvolvimento, da disposição e tamanho da equipe e até da metodologia de desenvolvimento utilizada. No caso de projeto de alta complexidade e previsão de longo período de desenvolvimento, com equipes grandes ou geograficamente

dispersas, o suporte computacional se torna indispensável seja qual for a fase do projeto. (Budgen, 2013) Nas fases do desenvolvimento onde a solução tecnológica já faz parte da discussão (ex. definição de linguagem de programação, banco de dados, etc.), o suporte computacional não é mais opcional. O suporte computacional ao desenvolvimento de software, composto por diversos tipos de ferramentas, é uma das camadas consideradas e estudas na Engenharia de Software. (Pressman, 2010, p.14)

Uso de notações pré-definidas. A atividade de modelagem no desenvolvimento de software envolve diversos padrões, linguagens ou notações pré-definidos que podem ser utilizandos conforme a necessidade do software em desenvolvimento. (Pressman, 2010) A *Unified Modeling Language* – UML (OMG, 2015) é a linguagem ou notação mais conhecida e usada no desenvolvimento de software. A UML dá suporte tanto a especificações de modelos mais abstratos, ainda mais próximos ao domínio (diagrama de casos de uso, de atividades), como também de modelos onde o domínio vai sendo organizado e pensado de acordo com a tecnologia escolhida para a construção do software e também para a geração automática de código de software (diagrama de classe, de dados). A UML oferece uma grande variedade de representações e recursos. No entanto, a forma como a UML será usada, como será apropriada pelos envolvidos no processo, deve ser definida particularmente para um determinado projeto ou equipe de desenvolvimento. (Dobing e Parsons, 2006) (Petre, 2013) A notação pré-definida é a 'língua' a ser utilizada na comunicação dentro do contexto de desenvolvimento de software. Contudo, que 'frases' ou 'textos' serão elaborados com essa 'língua', fica a cargo dos envolvidos (seus 'usuários').

Contexto de forte colaboração. O desenvolvimento de software é um processo onde, normalmente, um grupo de pessoas se coordena para desenvolver um único artefato: o software. Do momento da concepção e especificação do software até a sua entrega, quando existe um artefato computacional com o qual o usuário pode interagir, várias pessoas trabalham juntas, de forma colaborativa para viabilizar o resultado almejado. Para que esse esforço de várias pessoas tenha êxito, os aspectos de colaboração, coordenação e cooperação da atividade realizada devem ser considerados e apoiados. (Ellis e Gibbs, 1991) Num contexto colaborativo o

conhecimento mútuo (*mutual awareness*) e a consciência da atividade (*activity awareness*) (Carroll et al., 2009) são conceitos interessantes a se considerar no contexto do processo de desenvolvimento de software. O conhecimento mútuo está relacionado a saber o que os demais estão fazendo, o que fizeram, quais são suas capacidades e conhecimentos. Já a consciência da atividade refere-se a quando esse conhecimento mútuo é contextualizado para uma porção do processo colaborativo considerado. No caso da nossa pesquisa consideramos exclusivamente as atividades de modelagem de software. (Carroll et al., 2009) No desenvolvimento de software, temos as pessoas que realizam as atividades, aqueles que constroem e usam os modelos de software. Portanto, o conhecimento mútuo entre quem constrói e quem usa os modelos de software, bem como a consciência de ambos sobre a atividade de modelagem em si, são parâmetros importantes para modelagem de software. A construção e uso de modelos pode ser usada para a representação de um entendimento comum (*common ground*) (Canoe e Rosson, 2012) entre os envolvidos no desenvolvimento do software. Dentro desse contexto altamente colaborativo, cada um deve ter a noção do que 'acontece em sua volta', dos envolvidos no processo, de como o que alguém faz pode afetar outra pessoa ou alguma coisa, da relação entre as etapas do processo. Quando várias pessoas estão colaborando para um mesmo objetivo, não é viável que cada uma foque somente na sua parte, é necessária uma consciência do todo, das partes e como tais partes se relacionam.

Considerando as características descritas acima, discutiremos a seguir as atividades de modelagem de software em si que são o contexto particular desta pesquisa, tendo como referência os fatores da tripla TNP, e como tais fatores são identificados e relacionados nesse contexto. Destacaremos em *itálico* a associação dos pontos discutidos com os fatores da tripla TNP.

De forma geral, vamos referenciar essas atividades como produção e consumo de modelos de software. Para esta pesquisa, modelos e diagramas são, em alguns momentos, sinônimos. Isto ocorre quando fazemos referência à representação visual de alguma abstração relacionada ao software em desenvolvimento. Definimos o mnemônico 'Mnot' para identificar quando modelo faz referência a sua representação, sua notação. Quando o termo 'modelo' fizer referência ao seu significado, usamos o mnemônico 'Msem'.

A construção de um modelo de software é uma atividade intelectual que pode ser realizada por uma ou mais pessoas de forma colaborativa. Normalmente o desenvolvimento do software está associado a um problema identificado no domínio, para o qual o software pode ser definido como parte da solução. (Pressman, 2010).

Os modelos de software são utilizados em diferentes níveis de abstração durante o processo de desenvolvimento (Pressman, 2010). Modelos podem ser usados para apoiar discussão e solução de problemas, oferecendo visões diferentes (completas ou segmentadas) das partes relacionadas ao problema. Os modelos apresentam as informações de uma forma usável pelas pessoas, através da externalização do problema em uma representação visual. (Funt, 1995) Além disso, podem agrupar informações relacionadas, o que facilita no processo de solução de problemas e oferece mais uma camada de significado para a informação, através da atribuição de uma representação visual a uma determinada informação. (Larkin e Simon, 1995)

Para os pontos discutidos acima, temos indicações da necessidade de investigação de P da tripla *TNP, porém temos mais de um P interagindo ao longo do processo de desenvolvimento. Também temos N, quando consideramos a representação visual dos modelos.*

A UML é a linguagem de *design* de modelos de software. Quando a linguagem de *design* é compreendida de forma consciente, desenvolvida e aplicada, esta pode melhorar a atividade de criação, resultando em melhores interações com o produto desenvolvido. (Rheinfrank e Evenson, 1996) As linguagens de *design* têm um papel importante na expressão do desdobramento do significado. Elas são os meios através dos quais: (a) *Designer*s constroem significado em objetos; (b) Pessoas aprendem a interpretar e usar objetos; e (c) Objetos são assimilados nas experiências e atividades das pessoas. Linguagens de *design* podem ser usadas de forma mais eficiente quando o significado é visto não somente como um atributo embutido no objeto, mas como uma qualidade que o objeto possui de viabilizar o *sense making* de um grupo, a construção de significados sobre o contexto onde o objeto está inserido. (Rheinfrank e Evenson, 1996) No contexto de modelagem de software, os modelos são os objetos construídos utilizando a notação UML, e a relação entre os modelos, notação e as

pessoas que constroem e usam os modelos é bastante forte, para que os modelos exerçam o seu papel no processo de desenvolvimento de software.

Não existe um caminho direto entre a intenção do *designer* e seu produto (*i.e.* modelo, software). Conforme ele trabalha num problema, está continuamente construindo um caminho para a solução, formando novos valores e entendimentos à medida que avança no processo de construção. A possibilidade de o produto 'falar' por si ao longo de sua produção e consumo, dá oportunidade aos envolvidos de 'responder', indicando suas expectativas, alcançadas ou não. (Schön e Bennett, 1996) Os modelos de software oferecem essa oportunidade de 'conversa' com os envolvidos a respeito do que ele representa. Utilizando o modelo, os envolvidos no desenvolvimento do software vão representar seus entendimentos, discutir, procurar um consenso, voltar a discutir, achar outro consenso, até que o objetivo (i. e. conclusão de um modelo de dados, do projeto de interface, fechamento de versão parcial do software final e outros) seja alcançado.

Para os pontos discutidos acima, temos indicações da necessidade de investigação de N da tripla *TNP, porém considerando como N vai ser usada, temos que considerar P (envolvidos no desenvolvimento, designer das ferramentas de modelagem de software, dono do software, etc.) e também T, para algumas etapas.*

Em um cenário onde várias pessoas estão discutindo um mesmo artefato ou quando várias pessoas estão construindo seus próprios significados a partir de um mesmo artefato sem influência de outras pessoas, ambos comuns no processo de desenvolvimento de software, o mais razoável a se buscar é a convergência de interpretações, onde a mensagem passada pelo artefato seja recebida por todos de tal forma que suas interpretações possam convergir ao final do processo de significação. Em ambos os cenários, a semiose, processo de significação ou de produção de significados, deve ser considerado seja ele realizado em grupo (semiose coletiva) ou individualmente. O estabelecimento de um 'entendimento único', onde uma mesma interpretação é feita por diferentes pessoas é algo questionável, quando consideramos o conceito de semiose ilimitada de Peirce (1992-1998). O princípio da semiose ilimitada define que na comunicação, por exemplo, a semiose irá continuar enquanto as partes que se comunicam estão

ativamente engajadas na compreensão mútua. Portanto, a semiose continua indefinidamente, parando e sendo retomado conforme surjam necessidades ou oportunidades práticas no contexto do intérprete. Assim sendo, dado que os elementos do ‘contexto pessoal’ de cada um sempre afetam o ‘contexto coletivo’ em que as pessoas estão envolvidas, não há como garantir que todo mundo pare e fique definitivamente estacionado no mesmo exato ponto de semiose que os demais. Além disso, é fato que na dinâmica da evolução do processo de construção de software há revisões, ajustes e novos significados que se adicionam ao contexto global do desenvolvimento do artefato e assim torna a ideia de ‘entendimento comum’ um tanto frágil se formos testar exatamente o que significa.

A definição deste ‘entendimento comum’ no desenvolvimento de software é necessária, já que é um contexto onde todos colaboram para a produção de um mesmo artefato. A própria teoria Peirceana (Peirce, 1992-1998) nos apoia nesta definição postulando que embora o processo semiótico seja ilimitado (infinito se considerarmos a escala da cultura ou da civilização e indefinidamente longo se considerarmos a escala individual da vida de uma pessoa), ele passa por momentos de estabilização (e retomada) regulados pelos signos vigentes no contexto em que o intérprete está. Por exemplo, uma lei pode restringir definitivamente a progressão da semiose na direção de um limite, tanto como a falta de recursos (tempo, espaço, verbas, informações, etc.) pode restringir a progressão da semiose de uma comunidade. Na prática, então, o processo semiótico converge ‘em torno de’ signos majoritariamente consistentes com as restrições de contorno, o que se torna equivalente a este ‘entendimento comum’ e explica ocasionais desentendimentos ou inconsistências que, se procuradas, são encontradas no processo interpretativo de qualquer grupo.

Como representação deste ‘significado comum’, os modelos de software podem ser observados e investigados de duas formas:

1) **Modelo é o único canal de comunicação** sendo uma representação de significados concebida e apresentada pelo produtor do modelo ao consumidor do modelo, onde o modelo é o único canal de comunicação e, portanto, crucial para o sucesso da mensagem que o modelo pretende passar. A representação do modelo (Mnot) é o

único meio através do qual o produtor do modelo comunica o significado do modelo (Msem) ao consumidor**.**

2) **Modelo de software como objeto de discussão** sobre o que este representa. Este modelo seria, então, a representação de um entendimento coletivo, onde diversas pessoas discutiram e negociaram sobre esta representação (Mnot) e seu significado (Msem). O modelo é uma representação construída de forma colaborativa, onde os envolvidos compuseram significados e definiram como estes seriam representados de tal forma que fizesse sentido para todos. Todos os envolvidos são produtores do modelo e também podem ser consumidores do mesmo modelo. A representação do modelo (Mnot) é construída de forma coletiva e o significado do modelo (Msem) é acordado e compartilhado por todos os envolvidos em sua construção.

Modelos de software são artefatos intelectuais, que têm a criatividade como uma forte característica. O escopo de construção é mais amplo do que meramente o artefato. Engloba os valores e intenções do *designer*, neste caso o **produtor do modelo**, pressuposições e conhecimento sobre as pessoas e atividades relacionadas e, claro, os elementos disponíveis para a construção do artefato. (Carroll, 2012)

Para os pontos discutidos acima, temos indicações da necessidade de investigação de vários P's num contexto bem característicos de comunicação. Tendo os modelos de software como o artefato central, N é inerente e o uso (ou não) de T é relevante e pode ter grande influência sobre o entendimento do contexto como um todo.

Como indicado anteriormente, esta pesquisa tem como ***foco de investigação a produção e consumo de modelos com o apoio de ferramentas***, tendo a TNP como ferramenta de articulação onde T (tool) é sempre parte do contexto investigado.,

Podemos considerar a modelagem de software como um trabalho sobre o *continuum* expressivo, que é um dos tipos de trabalhos semióticos definidos por Eco (2000) dentro da Teoria de Produção Sígnica. Refere-se ao trabalho de

interpretar e produzir signos com um determinado objetivo. Uma comunicação significativa acontece quando um ser humano (*i.e.* um agente capaz de interpretar um código) compreende a mensagem enviada pelo transmissor (Eco, 1976) O requisito fundamental para a comunicação, de acordo com a Teoria de Produção Sígnica, é que a mensagem esteja codificada em um sistema de significação. A comunicação só acontece quando o intérprete reconhece os signos emitidos como parte de um código culturalmente estabelecido. A interpretabilidade de signos é fundamentalmente dependente do reconhecimento de que estes estão codificados, e de conhecimento do código utilizado para esta codificação. Existe um *continuum* de expressão, variando de palavras aos gestos, desenhos, sons, e etc., a partir dos quais o agente escolhe os que melhor transmitem conteúdo da mensagem. (Eco, 1976) Para a construção de um artefato de comunicação a escolha dos signos da representação de tal artefato deve ser feita considerando a mensagem que o artefato irá transmitir.

Durante a modelagem de software, e aqui consideramos a produção e consumo, que pode originar mudanças e evoluções no modelo, o produtor do modelo tem um conjunto pré-determinado de elementos (signos), definidos pela notação UML, porém não está, necessariamente, preso a este conjunto de signos, podendo assim dispor de uma coleção de signos pré-definidos e outros signos produzidos durante a evolução do trabalho. No contexto específico desta tese, interessam os sistemas de significação que são apoiados por ferramentas computacionais para a modelagem de software.

Um dos objetivos do trabalho sobre o *continuum* expressivo é construir a dimensão retórica do que está sendo produzido. A retórica refere-se à eficácia da semiose disparada na mente dos destinatários da comunicação feita pelos signos, cujo objetivo é o de estabelecer os procedimentos para que um signo possa dar origem a outro signo. (Peirce, 1992-1998) Para viabilizar o suporte à retórica realizada durante o desenvolvimento de software através do modelo, o produtor deste deve definir, e representar, signos que ofereçam elementos para que aquele modelo produzido (um signo composto por outros signos) possa ser objeto de significação para os consumidores do modelo. Portanto é interessante examinar, nas ferramentas computacionais de apoio à modelagem, os recursos que são oferecidos para a elaboração ou exploração desta dimensão retórica.

Ao longo da construção de um software, o desenvolvedor cria o seu discurso sobre este software utilizando vários sistemas de significação diferentes (complementares e suplementares), dos quais os modelos são um desses sistemas. A retórica deste discurso, neste caso, é uma escolha de *design* de comunicação multi-modal, multi-meios, multi-código, etc. Na medida em que estas escolhas são conscientes e (para fins práticos) sistemáticas, podemos dizer que o produtor deste discurso criou/usa uma retórica própria. Modelos de software são um de muitos elementos que podem integrar (ou não) esta escolha retórica.

Para o ponto de discussão apresentado acima, temos a presença potencial dos três fatores da tripla *TNP, apesar da T nem sempre ser utilizada. Porém, se considerarmos o conceito do continuum expressivo ao longo do processo de desenvolvimento de software, T, N e P relacionados vão ser parte do contexto investigado em algum ponto.*

Para investigar o contexto de interação com ferramentas de modelagem de software, a área de IHC oferece diversos métodos e técnicas. (Barbosa e Silva, 2010) A Engenharia Semiótica, em particular, apresenta métodos de avaliação que consideram, claro, o usuário (P), mas também incluem no contexto de avaliação o *designer* (P). Porém, outros P's não estavam considerados no escopo de avaliação. São os P's que identificamos no processo de desenvolvimento de software, pessoas que vão executar outras atividades ao longo do processo, após ou paralelamente, à modelagem de software. Outra perspectiva que precisou ser explicitada para ser analisada foi o usuário de ferramentas de modelagem de software como **produtor do modelo** e, em outros momentos, como **consumidor do modelo.** O modelo de software em si, como um artefato produzido por uma pessoa, para ser consumido por outras, também foi analisado separadamente, porém considerando T e N usados para a sua construção. Os P's considerados pela tripla TNP estão apresentados na Figura 10 e serão objetos de investigação e análise desta pesquisa. Isto corresponde a adicionar papéis ao usuário, o que na Engenharia Semiótica só é postulado para aplicações de *Groupware* (de Souza, 2005) (Prates, 1998) (Barbosa, 2006) e de *End User Development* (Sampaio, 2010) (Monteiro et al., 2013).

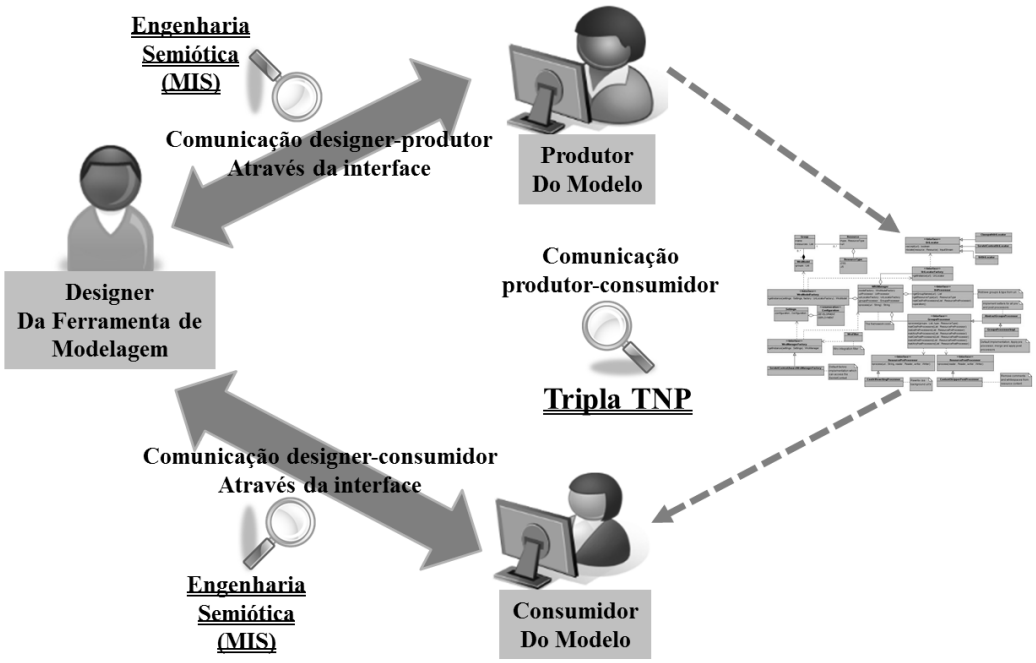


Figura 10. Os "Ps" investigados

3.2.2
Combined Semiotic-Cognitive Evaluation method (CSCE)

Tendo o foco desta pesquisa na investigação da influência da interação com ferramentas de modelagem sobre os modelos que com ela são produzidos ou consumidos, procuramos identificar métodos de avaliação de IHC que considerassem em sua investigação o espaço de interesse definido pela TNP. Para a realização desta pesquisa sentimos, então, a necessidade de desenvolver um método que combina duas das principais perspectivas de IHC, que apresentam profundidade comparável, porém contrastes significativos: Engenharia Cognitiva (Norman, 1986) e Engenharia Semiótica (de Souza, 2005).

A Engenharia Cognitiva, a perspectiva orientada para a usabilidade, e uma das mais conhecidas caracterizações teóricas da interação humano-computador, tem influenciado fortemente a abordagem de sistemas centrados no usuário (*User-Centered Design*). Ela foca na interação usuário-sistema. Quando define o fenômeno central de interesse em IHC, essa teoria, como o nome indica, foca somente no usuário, não levando em consideração nenhum dos demais perfis de pessoas que, de uma forma ou de outra, podem influenciar a maneira como usuário interage com artefatos computacionais (*i.e. designer*s e desenvolvedores dos sistemas, donos dos sistemas, etc.).

A Engenharia Semiótica, teoria base desta pesquisa e apresentada em mais detalhes na seção anterior, tem uma perspectiva orientada para a comunicação. Ela enfatiza a comunicação e processos de significação que ocorrem durante a interação, e traz o *designer* para o contexto da interação humano-computador. A interface é o preposto do *designer* durante a interação, portanto, mediando a comunicação *designer*-usuário. (de Souza, 2005)

A diferença mais relevante entre essas duas perspectivas de IHC é ***quais pessoas*** cada teoria leva em consideração. A perspectiva cognitiva foca no usuário e suas experiências durante a interação com a interface de sistemas. Embora, obviamente, saibamos que os sistemas são construídos por pessoas, a Engenharia Cognitiva mantém os produtores de software fora do cenário quando examina a interação. A perspectiva semiótica, ao contrário, considera produtores de software como parte do fenômeno interativo.

A combinação das perspectivas cognitiva (usabilidade) e semiótica (comunicabilidade) viabiliza uma inspeção mais abrangente de todas as partes relevantes dos objetos de análise e da relação entre eles. Para compor o ***Combined Semiotic-Cognitive Evaluation method*** **(CSCE)** (Figura 11), selecionamos o Método de Inspeção Semiótica - MIS (de Souza e Leitão, 2009), como o método da perspectiva semiótica, e o *Cognitive Dimensions of Notations framework* – CDNf (Blackwell e Green, 2003), como o método da perspectiva cognitiva. Ambos os métodos serão apresentados em detalhes mais à frente nesta seção.

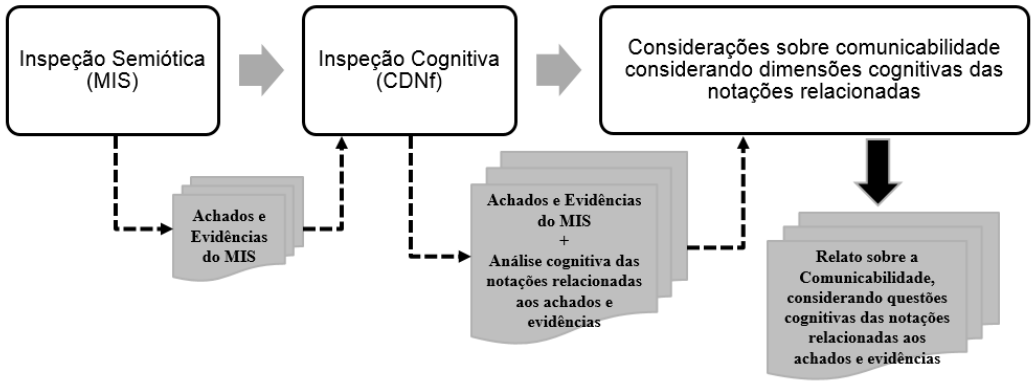


Figura 11. CSCE - Combined Semiotic-Cognitive Evaluation method

O CSCE é iniciado pela execução do MIS, originando assim um conjunto de achados e evidências relacionados a questões de comunicabilidade do sistema

avaliado. As evidências vão indicar porções da interface e outros elementos do sistema que ajudam na composição e ilustração de tais questões. Ao final da execução do MIS, temos um subconjunto dos signos do sistema, relacionados às evidências coletadas na inspeção. Este subconjunto de signos compõe o sistema de significação daquela porção inspecionada do sistema, é o conjunto de notações escolhidas e utilizadas pelo *designer* para construir sua mensagem para o usuário.

Após a execução do MIS, uma porção do sistema está delimitada, assim como o conjunto de notações relacionados às questões de comunicabilidade. Então, para cada evidência coletada no MIS, é feita a análise considerando as dimensões cognitivas definidas no CDNf. O objetivo desta etapa é associar às notações que caracterizam as questões de comunicabilidade, características cognitivas que possibilitem elaborar um relato mais completo e rico, contando com questões de comunicabilidade e, associadas a elas, discussões sobre as dimensões cognitivas das notações que compõem o sistema.

A última etapa do CSCE é a consolidação dos relatos sobre problemas de comunicabilidade identificados, relacionando a eles questões cognitivas das notações que colaboram para que tais problemas de comunicabilidade aconteçam. Um exemplo de relato de problema de comunicabilidade, com questões cognitivas associadas, poderia ser um pequeno ícone de ‘proibido’ aparecer rapidamente quando o usuário tenta realizar uma ação não permitida, porém é tão rápido, que o usuário pode não perceber e ficar tentando várias vezes a mesma ação não permitida. A notação escolhida pelo *designer* (ícone de ‘proibido’ piscando rápido) apresenta problemas em sua dimensão cognitiva de visibilidade. Não dá oportunidade ao usuário de ‘ver a mensagem do *designer’*. Uma solução possível seria apresentar tal mensagem em uma caixa de texto onde o usuário precisaria dar ‘Ok’ indicando um ‘Ah, tá! Entendi!’ através da interface.

A combinação do MIS e do CDNf tem sido desenvolvida e evoluída desde 2012 no contexto desta pesquisa de doutorado. Uma versão do método já foi aplicada para a avaliação de notações visuais de programação (Ferreira et al., 2012), de linguagens de *application programming interface* (API) (Afonso, et al., 2012), assim como ferramentas de modelagem (Ferreira e de Souza, 2013) (Ferreira et al., 2014).

A seguir, apresentamos em detalhes os dois métodos de avaliação que foram combinados para compor o CSCE: MIS e CDNf.

3.2.2.1
Método de Inspeção Semiótica (MIS)

O Método de Inspeção Semiótica (MIS) (de Souza e Leitão, 2009) é um método da Engenharia Semiótica que tem foco na emissão da metamensagem do *designer*-usuário. É executado por especialistas em Engenharia Semiótica que inspecionam a interface e analisam a interação, se colocando no lugar do usuário. O objetivo do método é reconstruir a mensagem de metacomunicação e, a partir dessa reconstrução, fazer uma análise sobre a comunicabilidade do software inspecionado. O MIS possui cinco (5) fases principais, conforme ilustrado na Figura 12. Na fase de preparação, o avaliador deve identificar perfis dos usuários a quem o sistema se destina e os objetivos que o sistema apoia, para então definir o escopo da avaliação (Barbosa e Silva, 2010). Para isso, o avaliador deve realizar uma inspeção informal nesta fase de preparação, buscando estabelecer o foco da análise (de Souza e Leitão, 2009).

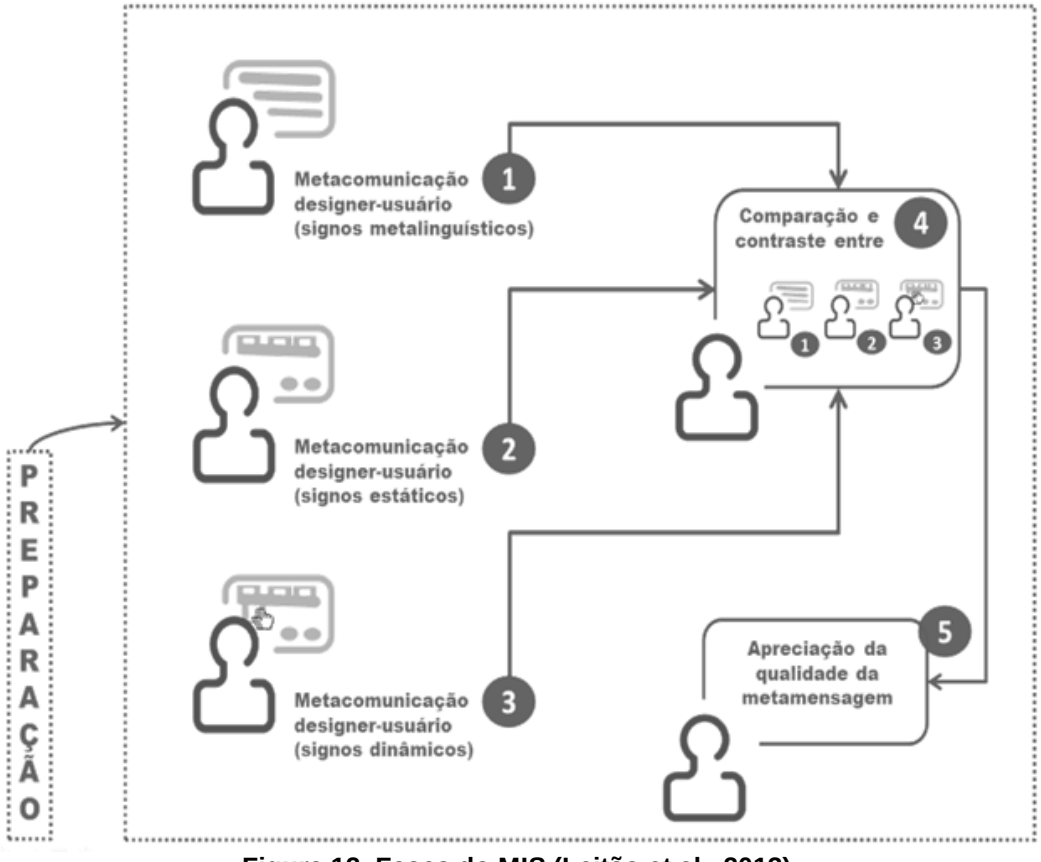


Figura 12. Fases do MIS (Leitão et al., 2013)

A mensagem de metacomunicação, o que se pretende reconstruir através da inspeção com o MIS, é transmitida pelos *designers* aos usuários através de signos

da interface do sistema que podem ser definidos em três classes (de Souza e Leitão, 2009): signos estáticos, signos dinâmicos e signos metalinguísticos.

Os signos estáticos têm o seu significado interpretado independentemente do tempo e de relações causais. O contexto de interpretação é limitado aos elementos que são apresentados na interface em um determinado instante no tempo, como por exemplo, barras de ferramenta e itens de menu.

Os signos dinâmicos surgem durante a interação e devem ser interpretados em relação ao desdobramento da interação no tempo. Eles são dependentes de relações causais como, por exemplo, a ação em consequência de clique de botão ou item de menu, ou a sequência de ações para realizar uma atividade (ex. Criar um arquivo). Outros exemplos são signos animados como a ampulheta de tempo e uma barra de progressão de um *download*.

Os signos metalinguísticos fazem referência aos signos estáticos, dinâmicos, e até mesmo metalinguísticos, tipicamente com o objetivo de explicar os outros signos. Geralmente eles se apresentam em forma de dicas, explicações, mensagens de erro, alertas e mecanismos de ajuda. Os signos metalinguísticos podem ser, eles mesmos, estáticos ou dinâmicos.

As atividades e tarefas realizadas na execução do MIS, estão apresentadas na Tabela 1 (Barbosa e Silva, 2010). As fases 1, 2 e 3 são fases análogas, onde são realizadas as atividades de coleta e interpretação de dados para cada um dos tipos de signo. A partir da inspeção de cada um dos tipos, é composto um *template* de metacomunicação. Após essa análise segmentada por tipo de signo, é realizada a comparação desses *templates*, procurando consistências entre cada signo, redundâncias e distribuição da metamensagem por tipo de signo.

Tabela 1. Atividades no Método de Inspeção Semiótica

inspeção semiótica	
atividade	**tarefa**
Preparação	▪ identificar os perfis de usuários ▪ identificar os objetivos apoiados pelo sistema ▪ definir as partes da interface que serão avaliadas ▪ escrever cenários de interação para guiar a avaliação
Coleta de dados	▪ inspecionar a interface simulando a interação descrita pelo cenário de interação
Interpretação	▪ analisar os signos metalinguísticos e reconstruir a metamensagem correspondente ▪ analisar os signos estáticos e reconstruir a metamensagem correspondente ▪ analisar os signos dinâmicos e reconstruir a metamensagem correspondente
Consolidação dos resultados	▪ contrastar e comparar as metamensagens reconstruídas nas análises de cada tipo de signo ▪ julgar os problemas de comunicabilidade encontrados
Relato dos resultados	▪ relatar a avaliação da comunicabilidade da solução de IHC, sob o ponto de vista do emissor da metamensagem

Ao final, é elaborada uma versão unificada da metamensagem, um *template* de metacomunicação preenchido com elementos coletados em todas as etapas anteriores e o avaliador apresenta as suas conclusões sobre comunicabilidade do sistema inspecionado, utilizando o que foi conhecido e evidenciado na análise segmentada e na comparação entre os *templates* de cada signo. Seu julgamento e conclusões devem ser pautados em evidências coletadas durante a inspeção.

3.2.2.2 Cognitive Dimensions of Notation framework (CDNf)

O *Cognitive Dimension of Notation framework* (CDNf) é um recurso analítico para apoiar as atividades de *design* e avaliação focando nas características da usabilidade de artefatos de informação, que normalmente são software. O CDNf pode ser aplicado para descobrir informações úteis sobre problemas de usabilidade. O CDNf pode ser usado tanto para a criação quando para a avaliação de interfaces de software e linguagens de programação usadas como artefatos de informação (Blackwell e Green, 2003). É um recurso bem flexível, podendo ser usado em diferentes situações, para notações mais visuais ou textuais, e não necessariamente considerando suporte computacional. (Clarke, 2001) (Kutar et al., 2002) (Petre, 2013a). O framework provê um vocabulário

comum para discussão de vários aspectos cognitivos de qualquer sistema de representação.

O sistema de representação de um software é composto por todos os elementos de interface e comportamentos associados que o *designer* do software decidiu utilizar quando o construiu, é a notação que o *designer* definiu para a construção do artefato. O objetivo do framework é melhorar a qualidade da discussão e decisões tanto na atividade de *design* e quanto na avaliação de sistemas de representação (Green e Blackwell, 1998). Há quatorze dimensões cognitivas definidas no CDNf, conforme apresentadas na Tabela 2:

Tabela 2. Lista das CDNs traduzida de Green e Blackwell, 1998

Cognitive Dimension of Notations (CDN)	Descrição
Abstração	Tipo e disponibilidade de mecanismos de abstração. Refere-se ao nível de abstração – agrupamento - imposto pela notação
Proximidade de mapeamento	Proximidade da representação com o domínio.
Consistência	Semânticas similares são apresentadas de uma maneira sintática similar.
Difusão	Complexidade ou verbosidade da notação em expressar significado.
Propensão a erros	Tendência da notação para induzir a erros e o sistema dá pouca proteção.
Operações mentais difíceis	Alta demanda de recursos cognitivos. Grau dos processos mentais necessários para os usuários entenderem a notação e para acompanhar o que está acontecendo.
Dependências ocultas	Relação entre duas entidades de tal forma que uma delas é dependente da outra, mas a dependência não é totalmente visível.
Comprometimento prematuro	Aplicação de decisões antes de possui a informação necessária para tal decisão e restrições de ordenação de tarefas.
Avaliação progressiva	Habilidade de avaliar o próprio trabalho em andamento a qualquer momento.
Provisoriedade	Flexibilidade da notação para usuários “brincarem com ideias”.

Cognitive Dimension of Notations (CDN)	Descrição
Expressão de papéis	O propósito de uma entidade e como esta entidade se relaciona com todo o componente são óbvios e pode ser inferido diretamente.
Notação secundária	Habilidade de usar notações além da sintaxe formal para expressar informação ou significado.
Viscosidade	Grau de esforço necessário para realizar uma mudança. Um sistema viscoso requer que o usuário realize muitas ações para alcançar uma meta.
Visibilidade	Habilidade de visualizar todos os componentes simultaneamente ou ver dois componentes relacionados lado a lado ao mesmo tempo.

O CDNf, apresenta cinco passos que orientam a análise de usabilidade de artefatos intelectuais utilizando as dimensões cognitivas: (1) Conhecer o sistema de representação; (2) Decidir o que o usuário irá fazer com a notação; (3) Escolher uma atividade representativa; (4) Para cada passo, se perguntar se o usuário pode escolher onde começar, como um erro será corrigido, se existirem dúvidas, que abstrações estão sendo usadas, e assim por diante. Esses passos vão gerar um perfil observado; e (5) Comparar o perfil observado com o perfil ideal para aquele tipo de atividade.

No caso do CSCE, os passos (1), (2) e (3) não são necessários, já que o MIS realiza as mesmas atividades na fase de Preparação. Como já está delimitada a porção da notação do sistema a ser avaliada no CSCE, partimos do passo (4) do CDNf para avaliar as dimensões cognitivas da notação do sistema inspecionado.

3.2.2.3
Escolha dos métodos para composição do CSCE

A escolha dos métodos de avaliação para compor o CSCE foi orientada pela relação de complementação e foco que cada um deles apresenta, oferecendo recursos para uma avaliação mais abrangente e rica. O MIS enquadra de forma completa o contexto da avaliação através do preenchimento do *template* de metacomunicação. Tendo esse *template* preenchido, o CDNf foca nas notações que compõem o *template*. Essas notações estão concentradas e duas porções do *template*, conforme destacado abaixo:

"Eis a minha visão de quem é você, ***que aprendi que você deseja ou precisa fazer, de que formas preferenciais*** *e por quê. Este é o sistema que consequentemente elaborei para você,* ***e esta é a forma como você pode ou deve usá-lo*** *para realizar um conjunto de objetivos que se enquadram nesta visão."*

Na porção "***que aprendi que você deseja ou precisa fazer, de que formas preferenciais"***, temos evidências fortemente relacionadas à notação escolhida pelo *designer* para construir sua mensagem, como por exemplo, itens de menu com opções de tarefas no sistema, a mesma opção no menu e em um botão, e outras notações apresentadas na interface avaliada. Na porção ***"e esta é a forma como você pode ou deve usá-lo"*** temos evidências, por exemplo, sobre o uso do teclado e do mouse para determinadas tarefas, ou a ação de criar novos objetos arrastando a partir de uma régua de objetos disponíveis. As evidências relacionadas a essas duas porções do *template*, apresentam elementos da notação do sistema, que podem ter características cognitivas que influenciam na comunicabilidade. Por isso, o CSCE complementa a análise semiótica, olhando as dimensões cognitivas das notações que compõem as evidências de tal análise.

3.2.3
Avaliação usando a tripla TNP e CSCE

A tripla TNP, o CSCE e a associação dos dois (CSCE+TNP) foram instrumentos de investigação criados para dar suporte a esta pesquisa científica. Não temos como objetivo que tais instrumentos sejam prontamente utilizados por outras pessoas. O perfil do pesquisador que possa ter oportunidade de se beneficiar do CSCE e da tripla TNP é bem específico. Precisa ser alguém com grande experiência em métodos qualitativos de pesquisa em IHC, também larga experiência na Engenharia Semiótica e seus métodos de avaliação, particularmente o MIS, e também precisa ter experiência com o CDNf. Outros pesquisadores podem utilizar a TNP e o CSCE contanto que estejam interessados em aprofundar a sua pesquisa ou realizar pesquisa correlata ou similar em outro domínio ou para outra finalidade.

A combinação do CSCE com a TNP apresenta cinco passos principais conforme apresentado na Figura 13. Os passos 1, 2, 3 e 4 foram executados para cada ferramenta avaliada. O passo 5 é a consolidação dos achados das avaliações

de todas as ferramentas. Cada passo será descrito a seguir, procurando indicar como inserimos a tripla na execução do CSCE, expandindo tanto o contexto avaliado quando o nosso vocabulário, para apresentar os resultados da avaliação considerando não somente as atividades de modelagem, mas o desenvolvimento de software de forma mais ampla.

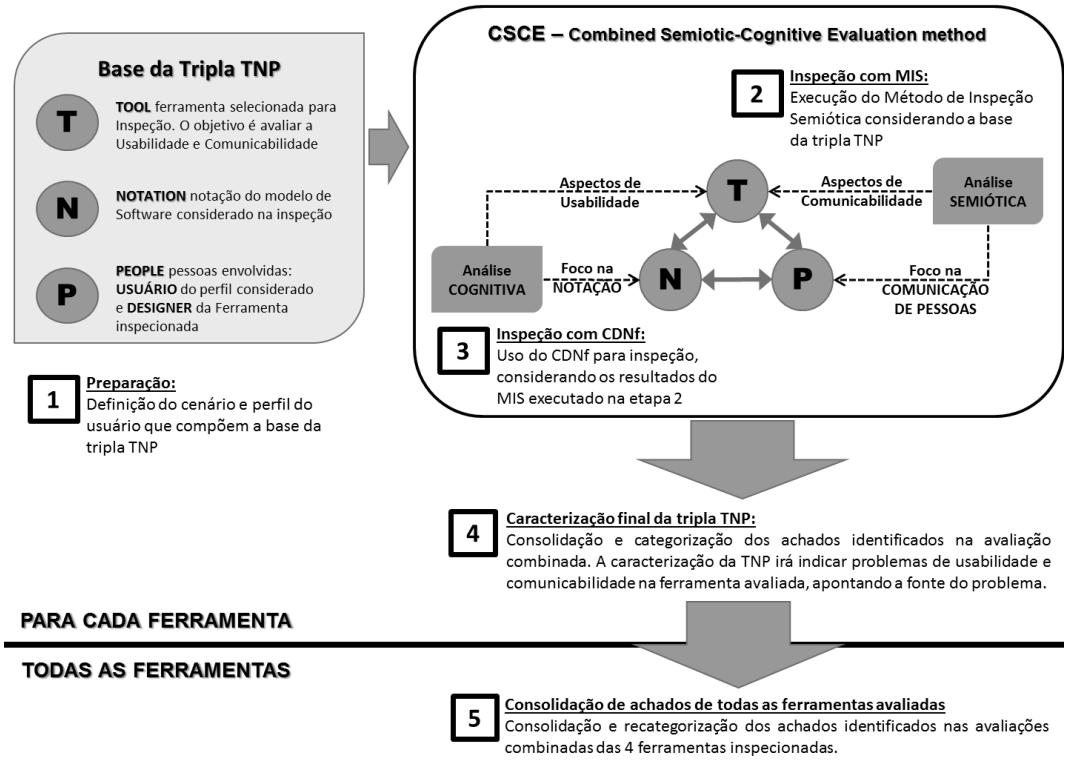


Figura 13. Avaliação CSCE + *TNP*

3.2.3.1 Preparação - Base da tripla TNP

Neste passo é definida a base da tripla TNP que é a referência sobre o perfil de usuário e cenário escolhidos para realizar a avaliação. Definimos qual será ferramenta avaliada (T), qual notação será o foco da avaliação (N), qual será o perfil de usuário considerado para a inspeção (P) e o cenário executado, que pode indicar outros P no contexto.

Antes da inspeção, a parte T da tripla ainda não está totalmente caracterizada, só é definida qual ferramenta será inspecionada. As partes N e P estão especificadas em termos gerais (no caso, N é a UML e P representa o usuário considerado, mas não sabemos nada sobre o outro P previsto na tripla, o

designer). O objetivo é que ao final da avaliação combinada, possamos caracterizar de forma mais completa a tripla para o cenário de inspeção específico em estudo.

3.2.3.2
Inspeção com MIS

Neste passo, a inspeção com o MIS é executada conforme foi definida e apresentada na seção anterior, porém a tripla será utilizada como referência. A ideia é identificar qual dos fatores, ou combinação de fatores, está associado ao problema de comunicabilidade identificado. Sendo o MIS um método da Engenharia Semiótica que tem foco na comunicação, espera-se que os problemas encontrados estejam concentrados na relação entre T e P, mas N pode impactar nesta relação. Portanto, os fatores T, N e P são observados em cada problema de comunicabilidade encontrado, e usados para apresentar as conclusões e considerações sobre a comunicabilidade da ferramenta inspecionada.

3.2.3.3
Inspeção com CDNf

Neste passo, a partir de cada problema de comunicabilidade identificado e apresentado na inspeção com o MIS, analisamos a notação relacionada ao problema reportado. Como o uso das CDNs, procuramos focar nas características cognitivas da notação relacionada ao problema e relacionar tais características ao problema de comunicabilidade identificado. O MIS nos oferece a oportunidade de identificar a relação entre o problema identificado na ferramenta (T) e a notação (N) e as CDN nos dão recursos para focar na notação em particular, possibilitando analisar qual o impacto cognitivo do uso de tal notação no problema de comunicabilidade identificado, que tem relação ou impacto nas pessoas (P) consideradas no cenário de avaliação.

3.2.3.4
Caracterização da tripla TNP

A tripla caracterizada irá indicar problemas de comunicabilidade e usabilidade na ferramenta avaliada e apontar a origem do problema, através da associação de cada problema aos fatores ou relações entre fatores da tripla. Por

exemplo, um problema pode estar associado somente à notação (N), outros podem estar associados aos papéis das pessoas na comunicação que acontece durante o desenvolvimento de software (P), ou associado à relação entre notação e ferramenta (T-N), à relação entre pessoas e ferramenta (T-P), ou aos três fatores da tripla (T-N-P).

A estrutura lógica da comunicação é identificada através da análise semiótica com o MIS, ao passo que a atividade mental realizada pelas pessoas quando usam a ferramenta é identificada pela análise cognitiva com o CDNf. A conclusão do método é uma rica caracterização de como a ferramenta apoia os usuários em atividades que estão, direta ou indiretamente, relacionadas ao cenário inspecionado.

3.2.3.5
Consolidação de Achados de Ferramentas Avaliadas

Após a avaliação combinada de cada ferramenta, os achados são consolidados e analisados, podendo ser necessária uma recategorização para a organização de todos os achados, gerando novas categorias e subcategorias relacionadas.

A definição das categorias é feita observando os fatores da TNP associados aos achados organizados naquela categoria. As subcategorias de uma categoria precisam apresentar pelo menos um fator da TNP comum a todas. Desta forma, este fator comum funciona como o balizador da característica constante daquela categoria, que pode ser somente um dos fatores (ex. P) ou uma relação de fatores (ex. T-N, T-N-P), agrupando as evidências das avaliações de todas as ferramentas sob uma mesma categorização.

3.3
Sobre este capítulo

Este capítulo detalhamos o ferramental metodológico (TNP e CSCE) desenvolvido como recurso essencial para a investigação dos problemas e questões de IHC enfrentadas por produtores e consumidores de modelos de software durante a interação com ferramentas de modelagem. Esses problemas e questões identificados são evidências e indícios de duas premissas consideradas nesta pesquisa:

1) É possível que, se os desenvolvedores de software tiverem problemas de IHC com suas próprias ferramentas, então a qualidade do seu produto pode ser impactada;
2) É possível que, se a qualidade do produto for impactada, então o usuário final experimente problemas de interação que estejam direta ou indiretamente ligados aos problemas de IHC experimentados pelos desenvolvedores.

A partir da análise e discussão de problemas de IHC, identificados com o ferramental, evidenciados no contexto de modelagem de software com o suporte de ferramentas, podemos caracterizar e organizar evidências para responder a questão de pesquisa: Como ferramentas e notações de modelagem influenciam e apoiam o processo intelectual de produção e consumo de modelos de software através da interação humano-computador?

4
Metodologia da pesquisa

Neste capítulo apresentamos a metodologia da pesquisa executada nesta tese[1], indicando a motivação para a escolha da abordagem de pesquisa e o perfil da pesquisadora. Por fim, detalhamos cada fase da metodologia da pesquisa, destacando suas etapas, insumos e produtos.

4.1
Abordagem de pesquisa

Esta pesquisa segue a abordagem qualitativa, que tem o significado como objeto de investigação. Métodos qualitativos buscam a exploração contextualizada de ações, comportamentos e processos interativos, com o objetivo de elaborar um quadro interpretativo que permita a análise e compreensão de um fenômeno que se apresenta como novo, imprevisível e irreplicável (Leitão, 2009). Métodos qualitativos são usados para explorar a evolução de significado ao invés de buscar uma definição ideal e definitiva do significado. (Creswell, 2013)

A ênfase dos métodos qualitativos está na análise focada e em profundidade da questão investigada, tendo como características principais o uso de pequenas populações ou amostras para observação. A combinação de técnicas e métodos de coleta e análise de dados, como também a multiplicidade de fontes de dados, busca a construção de uma visão holística da questão ou problema sob investigação. (Creswell, 2013)

Como visão de mundo, para usar os termos propostos por Creswell (2013), adotamos uma postura construtivista. O construtivismo rejeita a ideia de que o conhecimento científico possa ser separado de seu contexto humano. Em particular, advoga que o significado dos termos usados na teoria científica é construído socialmente, e que, portanto, interpretações do que a teoria significa são tão importantes quanto às observações empíricas para se julgar veracidade e

[1] O protocolo desta pesquisa foi formalmente avaliado pela Comissão de Ética em Pesquisa da PUC-Rio e aprovado no dia 9 de Novembro de 2014.

validade. O construtivismo se concentra menos em verificar teorias e mais no entendimento de como diferentes pessoas entendem o mundo e como essas pessoas associam significado a ações. Teorias podem emergir desse processo, mas estão sempre atreladas ao contexto em estudo. (Easterbrook e Singer, 2008) (Creswell, 2013)

O construtivismo apresenta pontos de contraposição à postura pós-positivista, mais tradicional entre pesquisadores das áreas exatas, muitas vezes chamado de "método científico" ou "pesquisa científica" (Tabela 3). O pós-positivismo prefere métodos iniciados a partir de teorias bem definidas das quais hipóteses verificáveis podem ser extraídas e testadas isoladamente. Essa postura é reducionista, no sentido de que estuda o todo através da quebra desse todo em componentes isolados e mais simples. Para seguir essa postura é preciso, evidentemente, determinar se o fenômeno de interesse pode ser isolado do contexto (Creswell, 2013). Este aspecto de isolamento é justamente uma das razões da adoção da postura construtivista na pesquisa aqui apresentada, já que o fenômeno em questão não poder ser isolado do contexto. Além disso, não buscamos verificar nenhuma hipótese pré-existente, estamos buscando o entendimento do fenômeno e, possivelmente, como resultado poderemos gerar novas hipóteses relacionadas ou evoluir teorias já conhecidas.

Tabela 3. Visão de mundo de pesquisa (Creswell, 2013)

Pós-positivismo	Construtivismo
• Determinístico • Reducionista • Observação empírica e medição • Verificação de teoria	• Entendimento • Significado de múltiplos participantes • Construção social e histórica • Geração de teoria

Também é de grande importância destacar que a Engenharia Semiótica (de Souza, 2005) é a teoria que embasa, direciona e inspira esta pesquisa. Considerando seus conceitos e seus métodos, todos de base qualitativa, (de Souza e Leitão, 2009) desenvolvemos esta pesquisa na área de IHC para caracterizar e compreender os fenômenos identificados quando pessoas interagem com tecnologia durante atividades do processo de desenvolvimento de software. IHC explora a natureza empírica dos métodos qualitativos para observar o uso da

tecnologia no contexto em que a mesma se aplica, utilizando métodos mais comuns das ciências sociais. (Blandford, 2014) Seu uso vem se apresentando como uma poderosa ferramenta para IHC, onde o contexto não pode ser desconsiderado. (Lazar et al., 2010)

Para realizar nossa investigação, lançamos uso de diversos métodos tanto de coleta quando de análise dos dados da pesquisa, buscando uma caracterização sócio-técnica rica e diversa. (Lazar et al., 2010) Aumentamos, assim, as chances de conhecer profundamente o fenômeno em questão e apresentar à comunidade científica nosso aprendizado e contribuições alcançadas. A metodologia desta pesquisa tem três grandes fases, que serão detalhadas a seguir neste capítulo, indicando todos os métodos utilizados, tipos de dados coletados e analisados, e seus achados associados.

4.2
Perfil da Pesquisadora

Sendo a pesquisa qualitativa fortemente baseada na análise e interpretação do pesquisador que a realiza, achamos importante apresentar o perfil da autora desta pesquisa. Oferecemos mais informações para que o leitor entenda o caminho percorrido e, talvez, boa parte da linha de raciocínio para as tomadas de decisão, assim como pode identificar algum viés que afeta a condução e as conclusões da pesquisa.

A pesquisadora pede licença aos leitores para falar em primeira pessoa de seu perfil, contando um pouco da sua história, profissional e acadêmica, que a levou a realizar essa pesquisa de doutorado:

"Me chamo Juliana Soares Jansen Ferreira e sou a pesquisadora principal do trabalho aqui apresentado. Minha experiência profissional e acadêmica tem grande influência na minha visão e postura como pesquisadora, por isso decidi me apresentar e contar um pouco da minha história. Formada como bacharel em informática, comecei a fazer parte do mercado de trabalho ainda na faculdade. Aprendendo a teoria e vendo a prática na área de desenvolvimento de software, sempre tive questionamentos sobre as práticas e dinâmicas das pessoas envolvidas nesse processo. Tive oportunidade de atuar em diversas posições neste contexto: como desenvolvedora, como analista de requisitos, como analista de negócios, e outras posições algumas vezes híbridas (desenvolvedor-analista).

Porém, a minha atuação, seja por oportunidade ou preferência, foi sempre focada na relação entre o domínio do problema e a solução tecnológica.

De 2002 até 2008, quando ingressei no mestrado, os modelos, sejam de software ou não, estiveram presentes na minha vida profissional como instrumentos poderosos de representação e comunicação entre pessoas. Combinando a minha experiência profissional com o conhecimento adquirido na graduação com ênfase de IHC, sempre observei o papel das ferramentas e notações de modelagem como partes fundamentais e altamente relevantes na construção e uso de modelos como objetos de comunicação.

No mestrado, quando estava profissionalmente envolvida com modelos de processo de negócio, a capacidade de comunicação desses e através desses modelos, foi tema da minha dissertação. Ver os modelos como objetos de representação do contexto de um problema que teria como solução um software, sempre me pareceu uma boa abordagem, já que antes de pensar na solução, o problema deve estar bem entendido por todos os envolvidos. Então, nada mais natural do que trazer essa problemática dos modelos para a pesquisa de doutorado, porém com um foco na relação entre pessoas e tecnologia na produção e uso dos modelos. O contexto de desenvolvimento de software foi uma escolha natural, pois sou parte deste contexto há pelo menos 12 anos. Podemos dizer que foi uma escolha um pouco egoísta, já que quero entender melhor e possivelmente colaborar com a solução de problemas que já me afetaram, e ainda afetam, e que também afetam meus colegas envolvidos no desenvolvimento de software. Espero que este breve perfil ajude você, leitor interessado no tema, no entendimento das minhas análises e decisões ao longo da pesquisa qualitativa apresentada nesta tese de doutorado. ”

4.3
Fases da Metodologia da Pesquisa

Esta pesquisa apresenta três grandes fases: Investigação do contexto de desenvolvimento de software, Investigação da Interação com Ferramentas de Modelagem e Triangulação de Resultados, como apresentado na **Erro! Fonte de referência não encontrada.**. Todas as fases da metodologia foram norteadas pela tripla TNP. Os três fatores foram sempre considerados tanto para a elaboração dos estudos quanto para a análise dos dados coletados.

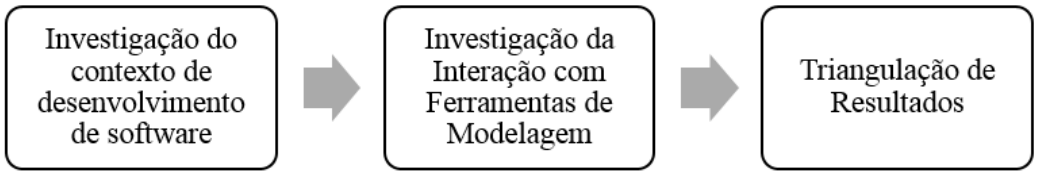


Figura 14. Metodologia da Pesquisa - visão geral

Na fase de Investigação do contexto de desenvolvimento de software foram aplicados questionários e realizadas entrevistas pós-questionário, contando com a colaboração de profissionais com experiência em desenvolvimento de software. Esta fase, que será detalhada como parte desta seção, tinha como objetivo coletar e analisar dados sobre as experiências desses profissionais em relação ao papel dos modelos de software, notações e ferramentas utilizadas para produção e consumo de tais modelos. O objetivo desta fase é entender o que desenvolvedores profissionais pensam acerca de modelos e do uso que eles (ou outras pessoas) fazem de modelos.

Na fase de Investigação da Interação com Ferramentas de Modelagem foram realizadas avaliações de comunicabilidade e usabilidade de ferramentas de modelagem de software através de métodos de inspeção, buscando conhecer como *designers* desse tipo de ferramenta se comunicam com seus usuários (desenvolvedores de software) através da interface e identificar as notações escolhidas por esses *designers* para construir tais mensagens aos seus usuários. Neste contexto, a notação da ferramenta é composta pela notação do modelo em si, neste caso UML, e as demais representações selecionadas pelo *designer* para construir significado na interface da ferramenta. Nesta fase utilizamos o CSCE, associado à TNP, para avaliar ferramentas de modelagem de software. Para esta fase optamos por construir a ferramenta de análise da interação (CSCE+TNP), entendendo que as ferramentas de análise de interação disponíveis, se utilizadas, não tratariam integradamente das relações entre os elementos contextuais da modelagem (TNP) e das duas dimensões (cognitiva e semiótica) inerentes à produção e consumo de modelos. O método de inspeção desenvolvido foi aplicado em profundidade na avaliação de 4 ferramentas de modelagem de software, identificando e caracterização problemas cognitivos e semióticos da interação de usuários com estas ferramentas.

Na fase de triangulação de resultados procuramos comparar, combinar e consolidar os achados das fases de Investigação do contexto de desenvolvimento de software (1), Investigação da Interação com Ferramentas de Modelagem (2) sobre questões ligadas aos três fatores da TNP. Essa triangulação seguiu duas linhas: (a) relacionar sistematicamente os problemas de interação encontrados em (2) com os achados de (1), estabelecendo relações entre interação com ferramentas e aspectos do processo de desenvolvimento de software (tal como reportados por desenvolvedores) e com isto enunciar a resposta solidamente fundamentada para a questão de pesquisa; e (b) contrastar a resposta e seus achados com achados de outros pesquisadores, reportados na literatura atual da área em que se situa a pesquisa. Nosso objetivo é não só identificar e caracterizar problemas que podem ajudar aos *designers* de ferramentas de modelagem a oferecer um suporte mais eficiente os seus usuários, mas também identificar e caracterizar problemas que apesar de poderem impactar o uso de tais ferramentas, podem estar também relacionados aos demais fatores (notação e pessoa). O foco desta fase é olhar o contexto como um todo e identificar problemas e questões relacionadas aos fatores da TNP, que podem auxiliar no entendimento de problemas e soluções relativos à construção e uso de modelos no processo de desenvolvimento de software. Além disso, contextualizar tais achados com o que outros pesquisadores já reportaram sobre o assunto.

4.3.1
Investigação do contexto de desenvolvimento de software

Na fase de investigação do contexto de desenvolvimento de software, procuramos coletar e analisar dados sobre as experiências de profissionais de desenvolvimento de software em relação ao papel dos modelos de software, notações e ferramentas utilizadas para produção e consumo de tais modelos. O objetivo desta fase é entender o que desenvolvedores profissionais pensam acerca de modelos e do uso que eles (ou outras pessoas) fazem de modelos. Os relatos e experiências desses profissionais foram coletados através de um questionário e entrevista subsequente (entrevista pós-questionário). A maior parte das etapas desta fase foi guiada pelo conceito da TNP, considerando seus três fatores e suas relações, conforme indicado na Figura 14 com o signo  . Somente a etapa de

coleta de dados do questionário não considerou a TNP, já que foi feito de forma automática utilizando o *Google Forms*[2]. Os achados e resultados desta fase foram utilizados com referência para definição dos cenários da fase de Investigação da interação com ferramentas de modelagem de software. A seguir, detalhamos cada uma das etapas da fase de Investigação do contexto de desenvolvimento de software indicadas na Figura 15, seus insumos e produtos.

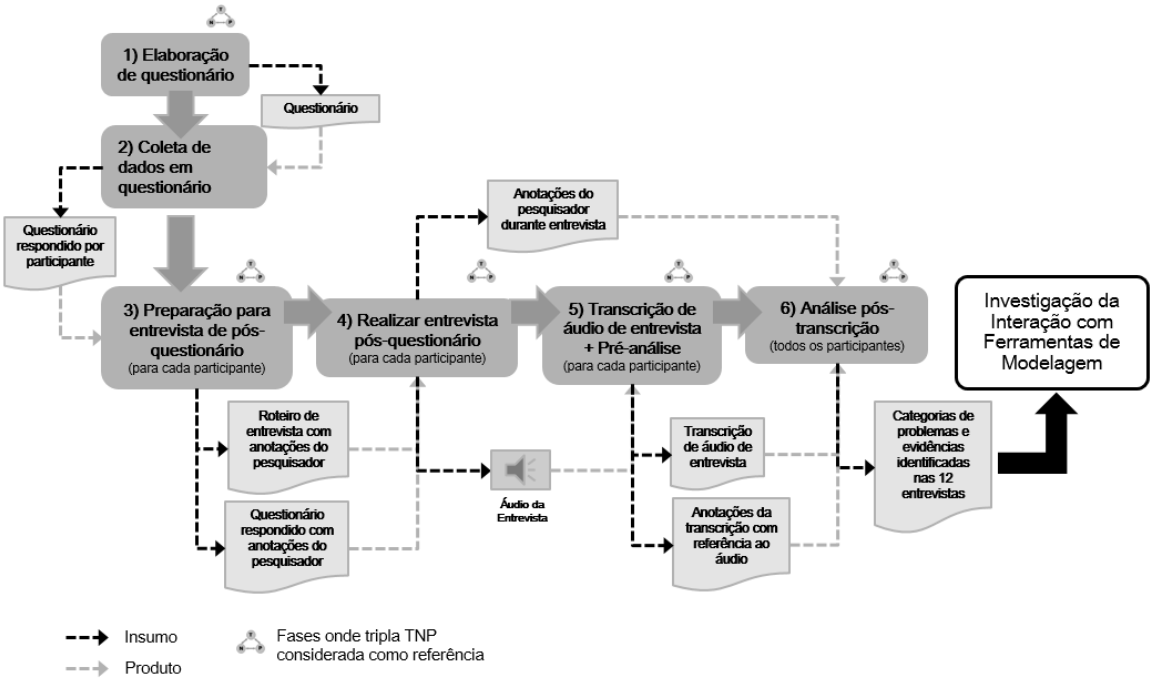


Figura 15. Metodologia – Primeira Fase

1) Elaboração de questionário:

Nesta etapa foi elaborado um questionário para ser aplicado em um conjunto diverso de profissionais com experiência em desenvolvimento de software. Os temas abordados no questionário buscavam traçar o perfil do profissional, com quais modelos de software tinha experiência de produção e consumo, quais ferramentas de modelagem conhecia, quais suas intenções típicas em relação ao modelo de software, suas percepções sobre os momentos de produção e consumo de modelos, sobre ações perante problemas de modelagem, seu relato de experiências com a UML e outras notações. Os objetivos desse questionário, que era composto de 15 perguntas (algumas delas condicionadas entre si), eram coletar dados sobre produção e uso de modelos no desenvolvimento de software e recrutar participantes para a fase seguinte, que consistia em uma entrevista sobre

[2] http://www.google.com/forms/about/

as respostas do questionário. O questionário utilizado está apresentado no Apêndice A, e com ele o termo de consentimento para a coleta dos dados do questionário relacionado.

2) Coleta de dados em questionário:

A etapa de coleta de dados dos questionários foi feita de forma automatizada utilizando o *Google Forms* como ferramenta. Primeiramente, a pesquisadora contatou profissionais conhecidos por ela (antigos colegas de trabalho, conhecidos de projetos realizados anteriormente, colegas de colegas indicados), procurando pessoas com um tempo razoável de experiência com desenvolvimento de software (pelo menos 5 anos de experiência) que desejassem participar da pesquisa. Uma vez que o profissional concordasse em participar, era enviado a ele o endereço eletrônico do questionário no *Google Forms*, que coletou automaticamente as respostas dos profissionais respondentes.

3) Preparação para entrevista pós-questionário

Esta etapa foi executada para cada um dos profissionais que respondeu o questionário e concordou em participar da entrevista. A última pergunta do questionário era sobre o desejo do profissional respondente em participar de uma entrevista sobre as respostas do questionário. Aqueles que responderam ‘sim’ se tornaram participantes dos estudos seguintes.

Para a entrevista com cada participante, foi feita uma preparação individual onde a pesquisadora analisou as respostas do seu respectivo questionário, fazendo anotações e marcações sobre pontos que necessitavam de esclarecimentos ou mais detalhes. Também foram identificadas oportunidades de aprofundar itens relacionados aos pontos marcados e anotados no questionário. O roteiro geral das entrevistas consta do Apêndice B.Após esta fase, a pesquisadora tinha em mãos o questionário respondido pelo participante, com suas anotações para a entrevista, e também o roteiro de entrevista personalizado para o participante, considerando suas respostas às perguntas do questionário.

4) Realização de entrevista pós-questionário

As entrevistas de pós-questionário foram agendadas junto aos profissionais que indicaram estar dispostos a participar. Uma vez escolhida a melhor data e horário para a entrevista, ela que foi realizada remotamente utilizando *Google*

Hangout[3] ou *Skype*[4]. Antes da entrevista ser realizada, a pesquisadora enviou aos participantes o termo de consentimento para sua realização (Apêndice C), esclarecendo sobre os dados coletados e o seu uso.

Durante a entrevista, a pesquisadora utilizou um gravador de áudio para coletar o diálogo entre o participante e a pesquisadora, além de realizar anotações sobre os pontos discutidos na entrevista.

5) Transcrição de áudio de entrevista + Pré-análise

Após a realização das entrevistas, elas foram todas transcritas. Durante a transcrição, a pesquisadora realizou uma pré-análise dos dados, registrando achados preliminares. A meta da pré-análise foi identificar pontos interessantes e serem aprofundados na fase de análise após a transcrição de todas as entrevistas. A pré-análise tinha um foco mais local (na entrevista sendo transcrita) para dar insumo para a análise final que tem um foco mais global (considerando todas as entrevistas).

Na pré-análise foi realizada uma Análise de Discurso, que é o estudo do uso da linguagem, em um determinado contexto, ricamente caracterizado em relação ao sujeito deste discurso. A análise procura por significado como um integrador na forma de o sujeito do discurso falar (informar), fazer (ação) e ser (identidade), e a gramática como um conjunto de ferramentas para realizar esta integração. (Gee, 2005)

6) Análise pós-transcrição

Após realizar a transcrição de todas as entrevistas e de posse dos dados da pré-análise, a pesquisadora realizou a análise pós-transcrição que teve como objetivo comparar, consolidar e categorizar ou fazer a codificação (Lazar et al., 2010) (Blandford, 2014) dos dados coletados. Esta análise teve como objetivo esmiuçar detalhadamente várias relações e implicações, internas ou externas, no discurso dos profissionais sobre modelos no desenvolvimento de software. Decidimos diversificar nossas fontes de dados (*i.e.* relatos de 12 profissionais) e manter um único analista, pois o referencial ou guia daquilo que tem de ser esmiuçado e aprofundado está constantemente presente (e unificado) na mente de

[3] http://www.google.com/+/learnmore/hangouts/?hl=pt-BR
[4] http://www.skype.com/pt-br/

um único intérprete. As categorias, questões e problemas identificados nesta etapa, foram usados na fase seguinte da metodologia. Para esta etapa também foi empregada a Análise de Discurso (Gee, 2005) como instrumento de investigação.

4.3.2
Investigação da Interação com Ferramentas de Modelagem

Na fase de investigação da interação com as ferramentas de modelagem de software procuramos conhecer como *designers* de ferramentas de modelagem de software se comunicam com seus usuários (desenvolvedores de software) e identificar as notações escolhidas por esses *designers* para construir tais mensagens destinadas aos seus usuários. Nesta fase foram realizadas avaliações de comunicabilidade e usabilidade de ferramentas de modelagem de software utilizando o CSCE, associado com a tripla TNP, ambos descritos em detalhe no capítulo anterior. Após a realização das avaliações das 4 ferramentas foi feita uma consolidação dos achados de todas as avaliações, reorganizando as categorias originalmente criadas nas avaliações individuais. As etapas desta fase estão apresentadas na Figura 16 e serão detalhadas a seguir.

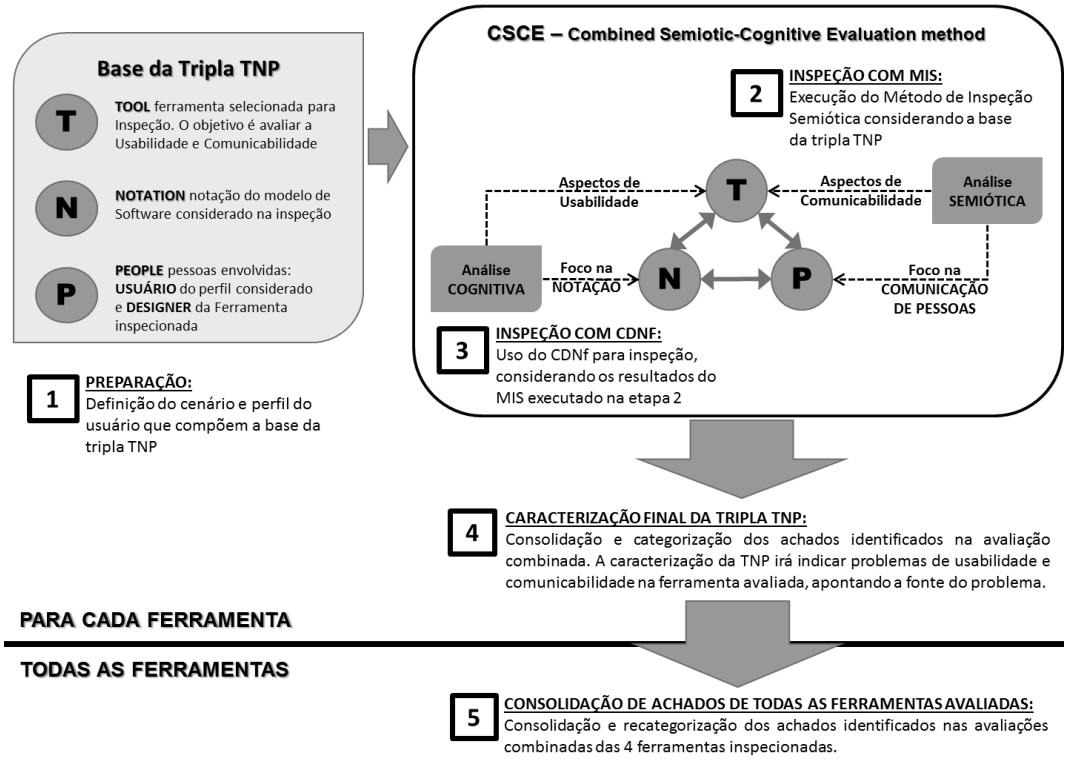


Figura 16. Metodologia - Fase 2

1) Preparação - Base da tripla TNP

A tripla TNP foi articulada para contribuir com a avaliação combinada como uma referência do contexto no qual a ferramenta de modelagem será inspecionada. Antes da inspeção, a parte T da tripla ainda não pode estar totalmente caracterizada. Só se define qual ferramenta será inspecionada. As partes N e P também estão especificadas apenas em termos gerais (no caso, N é a UML e P representa um profissional de desenvolvimento de software). Essa é a base da tripla TNP para a inspeção. Ao final da avaliação combinada, poderemos caracterizar de forma mais completa a tripla para o cenário de inspeção específico em estudo.

Para a definição da base da tripla TNP, utilizamos os achados da fase de Investigação do contexto de desenvolvimento de software. A partir do que aprendemos na primeira fase da pesquisa, foi possível identificar um cenário interessante para a investigação da interação com ferramentas de modelagem de software.

2) Inspeção com MIS

A inspeção semiótica com o MIS tem foco nas questões de comunicabilidade da ferramenta inspecionada. O MIS foi executado conforme concebido originalmente (De Souza e Leitão, 2009), como mostrado na Figura 17. Porém, os três fatores da TNP são considerados na fase final da inspeção semiótica, onde a qualidade da metamensagem é relatada pelo avaliador.

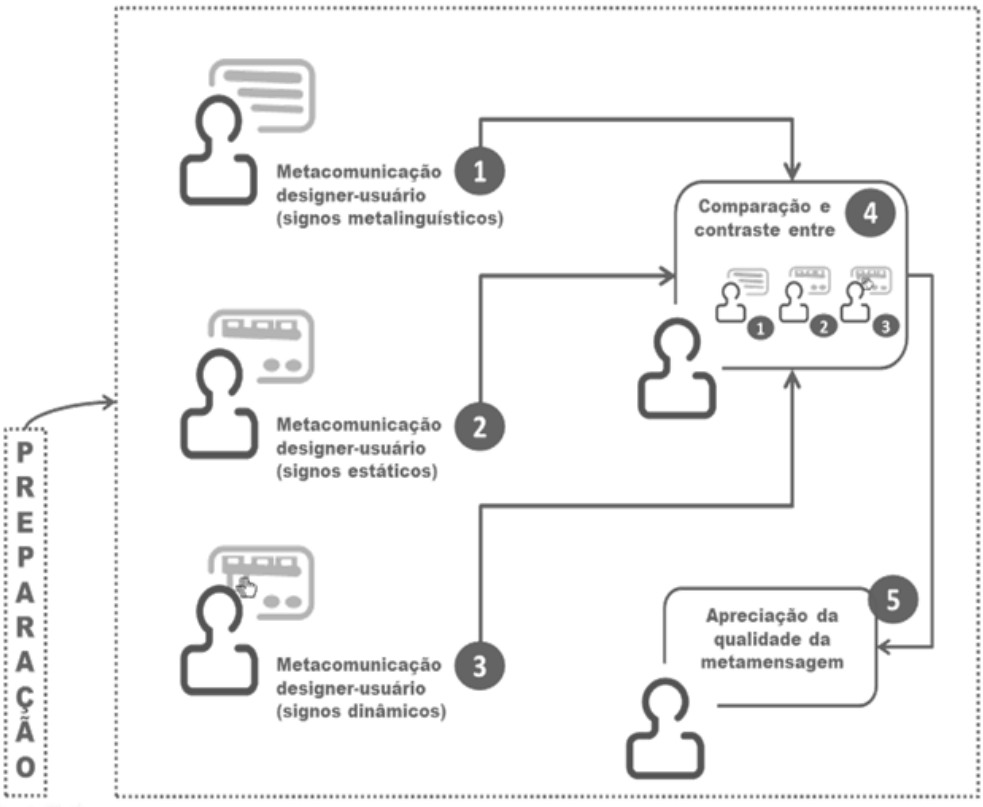


Figura 17. Etapas do MIS (Leitão et al., 2013)

A contextualização que a TNP proporciona à avaliação apoia o avaliador no relato das questões relacionadas à comunicabilidade da ferramenta inspecionada. Tendo a TNP como guia, o avaliador pode identificar achados além daqueles relacionados à ferramenta em si e ao momento de interação do usuário com a ferramenta. A intenção com o uso da ferramenta, conceito forte na Engenharia Semiótica, ganha uma abstração maior, dentro do contexto delimitado pela TNP.

3) Inspeção com CDNf

A inspeção cognitiva utilizando as CDNs tem como foco as questões de usabilidade da ferramenta inspecionada. Considerando as evidências destacadas durante a execução do MIS, ilustrando uma questão ou problema de comunicabilidade, as dimensões cognitivas da notação (CDN) apontam para questões de usabilidade das notações relacionadas à questão ou problema de comunicabilidade. Com o MIS observamos as notações combinadas num contexto de aplicação, de uso, e com as CDNs podemos destacar qual dimensão cognitiva de qual notação utilizada naquela evidência tem relação com a questão ou problema de comunicabilidade identificada.

4) Caracterização final da tripla TNP

A caracterização final da tripla TNP irá indicar questões de usabilidade e comunicabilidade na ferramenta avaliada e apontar a fonte do problema. Por exemplo, um problema pode estar relacionado somente à notação, enquanto outros podem estar relacionados ao papel das pessoas nas comunicações durante o processo de desenvolvimento de software, ou à relação entre notação e ferramenta (T-N), ou à relação entre pessoas e ferramenta (T-P) ou qualquer outro elemento ou relação da tripla.

5) Consolidação de achados de todas as ferramentas avaliadas

Após a avaliação combinada de cada ferramenta, os achados foram consolidados e analisados, sendo necessária uma recategorização para organização de todos os achados, gerando novas categorias com subcategorias relacionadas. As evidências para cada categoria e subcategoria foram escolhidas entre aquelas coletadas nas avaliações das ferramentas, buscando as que melhor ilustravam o problema ou questão relacionados.

4.3.3 Triangulação dos Resultados

Na fase de triangulação de resultados seguiu duas linhas: (a) relacionar sistematicamente os problemas de interação encontrados na fase de investigação da interação com ferramentas de modelagem de software com os achados na fase de investigação do contexto de desenvolvimento de software, estabelecendo relações entre interação com ferramentas e aspectos do processo de desenvolvimento de software (tal como reportados por desenvolvedores) e com isto enunciar a resposta solidamente fundamentada para a questão de pesquisa; e (b) contrastar a resposta e seus achados com achados de outros pesquisadores, reportados na literatura atual da área em que se situa a pesquisa.

A identificação de contradição entre os dados das fases de investigação do contexto de desenvolvimento de software, considerando os dados relacionados ao uso de ferramentas, e Investigação da interação com ferramentas de modelagem de software de ferramentas de modelagem de software pode indicar aos *designers* de ferramentas de modelagem de software características da ferramenta que

apresentam potenciais não intencionados pelo *designer*, ou características que não estão claras para o usuário.

Considerando que as duas fases iniciais da metodologia tiveram níveis de investigação diferentes, uma fase considerou um contexto com e sem ferramentas enquanto na outra fase foram realizadas avaliações diretas de ferramentas de modelagem de software; a ausência de relação de dados da primeira fase com a segunda é esperada. Porém, entendemos que iniciar a investigação a partir de um contexto mais amplo nos possibilitou fazer questionamentos sobre as atividades de modelagem de software que podem ser úteis para a melhoria ou expansão de ferramentas que apoiam tais atividades.

Com o contraste dos achados desta pesquisa com dados de pesquisas realizadas anteriormente por outros pesquisadores foi possível, por exemplo, reforçar evidências identificadas, indicando possíveis caminhos de aprofundamento sobre características problemáticas de ferramentas de modelagem no contexto de desenvolvimento de software.

Como todas as etapas, a triangulação também considera a tripla TNP como referência, portanto as questões e problemas são contextualizados por esses três fatores, possibilitando a identificação, por exemplo, do que pode ser resolvido ou minimizado com a ajuda da ferramenta, ou o que está num escopo maior, que a ferramenta por si só não pode solucionar isoladamente como, por exemplo, questões relacionadas às pessoas envolvidas no processo de desenvolvimento.

4.4
Sobre este capítulo

A questão de pesquisa determinou a definição desta metodologia, buscando diferentes perspectivas e fatores para responder: Como ferramentas e notações de modelagem influenciam e apoiam o processo intelectual de produção e consumo de modelos de software através da interação humano-computador?

Na investigação do contexto de desenvolvimento de software, usando a TNP como guia, foi possível investigar ferramentas, notações e práticas do processo de produção e consumo de modelos de software (*i.e.* como pessoas usam notações e ferramentas ao longo deste processo). Na investigação da interação com ferramentas de modelagem de software, utilizando o CSCE associado à tripla TNP, avaliamos as ferramentas de modelagem que podem apoiar a produção e o

consumo de modelos de software. Nesta fase inserimos o *designer* das ferramentas de modelagem no contexto investigado (*i.e. designers* fazem parte do fator PESSOAS da TNP). Ao final, com a triangulação, foi possível caracterizar problemas e questões comparando e confrontando os achados de cada uma das fases da metodologia. Na triangulação buscamos evidências de das duas fases anteriores que complementassem ou confirmassem algum problema ou questão identificada, ou que se relacionassem (ex. decisão de não usar ferramentas de modelagem, identificado na fase Investigação do contexto de desenvolvimento de software relacionada à limitação da mesma, identificada na fase de investigação da interação). Através da investigação dos fatores da TNP e suas relações, o processo de produção e consumo de modelos foi posicionado como parte de um processo maior, o de desenvolvimento de software. Desta forma, ampliamos nosso contexto de investigação, possibilitando a identificação de problemas e questões de diferentes níveis relacionados à TNP. Neste contexto ampliado, pudemos discutir a comunicabilidade dos modelos de software, e como notações e ferramentas podem influenciar ou apoiar essa comunicabilidade ao longo do processo de desenvolvimento de software. Os resultados ainda foram enriquecidos com a análise e discussão considerando pesquisas anteriores realizadas por outros pesquisadores sobre desenvolvimento de software, com foco nos modelos e ferramentas de modelagem.

5
Resultados dos estudos realizados

Neste capítulo detalhamos os estudos realizados no contexto desta tese, assim como apresentamos os achados relacionados a cada fase executada. Apresentamos os achados mais relevantes, que ilustram situações, problemas e questões interessantes para discussão nesta pesquisa.

A seguir, apresentamos os achados da fase de Investigação do contexto de desenvolvimento de software, onde utilizamos questionário e entrevista para coletar dados sobre a experiência de profissionais de desenvolvimento de software acerca dos modelos de software, ferramentas e notações utilizadas. Esses achados foram analisados e categorizados tendo a tripla TNP como referência.

Após a apresentação dos achados sobre o contexto de desenvolvimento de software, apresentamos os achados da investigação da interação com ferramentas de modelagem de software, coletados através da avaliação semiótica-cognitiva de 4 (quatro) ferramentas de modelagem de software reconhecidas no mercado, utilizando o CSCE e a tripla TNP para caracterizar a interação considerando ferramenta, notação e pessoas no processo de produção e consumo de modelos de software.

Por fim, realizamos a triangulação dos achados nas fases de investigação do contexto de desenvolvimento de software e da investigação da interação com ferramentas de modelagem, procurando identificar problemas e questões relacionados e observados nas duas fases. Além disso, discutimos problemas e questões identificados na investigação do contexto de desenvolvimento de software que podem indicar caminhos valiosos de investigação para aprimoramento das ferramentas de modelagem de software ou de outros fatores relacionados ao contexto de modelagem e de desenvolvimento de software. Os achados das fases de investigação também foram triangulados com achados de outras pesquisas relacionadas ao tema de modelagem de software.

5.1
Investigação do contexto de desenvolvimento de software

Na fase de investigação do contexto de desenvolvimento de software, procuramos levantar e coletar dados sobre experiências de profissionais de desenvolvimento de software relacionadas aos modelos de software. Buscamos identificar situações de produção e consumo dos modelos, considerando ferramentas e notações utilizadas, pessoas envolvidas, casos de sucesso e fracasso, ações de contorno relacionadas aos modelos e outros relatos sobre o papel dos modelos no contexto de desenvolvimento de software. Nesta fase definimos um contexto mais amplo para a nossa investigação, onde buscamos levantar dados sobre experiências com e sem o uso de ferramentas de modelagem de software. Nosso objetivo foi explorar as razões pelas quais profissionais decidem em usar ou não ferramentas, qual a relação entre essa decisão e o contexto do projeto de desenvolvimento de software, o que a modelagem sem as ferramentas pode indicar como potenciais melhorias para as mesmas e outras questões relacionadas ao uso ou não de ferramentas de modelagem que possivelmente seriam levantadas pelos participantes. Esta fase está detalhada na seção 4.3.1 do capítulo 4 desta tese.

Obtivemos retorno de 31 (trinta e um) questionários respondidos, sendo que 1 (um) foi desconsiderado por não haver a indicação do consentimento para uso dos dados coletados. Dos respondentes considerados, 12 (doze) concordaram em participar da entrevista pós-questionário, se tornando, portanto, participantes desta pesquisa. A maior parte dos participantes (10/12) tem mais de 5 anos de experiência com desenvolvimento de software e os perfis de atuação em projetos ficaram distribuídos entre profissionais que atuaram principalmente na etapa de especificação do software (5/12), etapa de desenvolvimento (5/12) e na parte de banco de dados (2/12). Todos os participantes indicaram experiência no uso de modelos e a maior parte dos participantes indicou também experiência na construção modelos de software (10/12), tendo os demais (2/12) somente experiência acadêmica de construção de modelos (Tabela 4). O questionário e termo de consentimento relacionado estão apresentados no Apêndice A.

Tabela 4. Consolidação dos números dos questionários respondidos

Totalizadores de Respostas do questionário	**Nº**
Total de questionários respondidos	31
Total de questionários respondidos e considerados como dados de pesquisa	30
Total de Participantes (respondentes que concordaram em participar da entrevista pós-questionário)	12
Participantes com mais de 5 anos de experiência com desenvolvimento de software	10
Participantes que atuaram principalmente na etapa de especificação do software	5
Participantes que atuaram principalmente na etapa de desenvolvimento do software	5
Participantes que atuaram na parte de banco de dados do software	2
Participantes indicaram experiência no uso de modelos	12
Participantes indicaram experiência na construção de modelos	10

Para a entrevista, elaboramos um roteiro que, junto com as respostas do questionário do participante entrevistado, foram usadas para guiar cada uma das entrevistas individuais, aprofundando alguns pontos das respostas anteriores e outros pontos complementares sobre a experiência com modelos de software. Houve a necessidade de ajuste no roteiro após a terceira entrevista, devido ao relato do Participante 3 estar fortemente relacionado às ferramentas de modelagem, o que não foi identificado nas entrevistas 1 e 2 (detalhes a seguir).

O primeiro roteiro, utilizado nas entrevistas 1, 2 e 3, não contava com perguntas específicas sobre ferramentas utilizadas na construção e uso de modelos. A necessidade de tais perguntas foi identificada durante a entrevista com o Participante 3 que citou constantemente ferramentas de modelagem em seus relatos. Este participante relatou experiências com a abordagem *model-driven*, onde as ferramentas tem um papel chave na construção e transformação dos modelos em código-fonte. A forte presença de ferramentas de modelagem no relato das entrevistas não ocorreu com os Participantes 1 e 2 porque ambos

relataram experiências de construção e uso de modelos com pouco ou quase nenhum uso de ferramentas, sendo usados outros recursos como quadro branco ou mesmo papel e lápis. Portanto, incluímos no roteiro perguntas ou pontos de discussão sobre as ferramentas que apoiam a construção e uso de modelos de software e retornamos ao Participante 3 para complementar a entrevista. O retorno não foi necessário aos Participantes 1 e 2 já que os relatos feitos nas entrevistas indicavam a ausência de ferramentas para interagir com os modelos de software. Para os Participantes 4 até 12, o novo roteiro foi utilizado. Os roteiros utilizados nas entrevistas e o termo de consentimento relacionado estão apresentados na íntegra em Apêndice B e Apêndice C, respectivamente.

Os dados coletados e analisados nesta fase foram as respostas do questionário, tanto daqueles que somente responderam o questionário online (Respondentes) quanto dos que se tornaram efetivamente Participantes do estudo, consentindo e participando da entrevista. As respostas dos Participantes foram utilizadas na preparação de suas respectivas entrevistas com cada um dos participantes; já as dos demais Respondentes foram utilizadas na fase de análise, categorização e triangulação dos dados coletados nas entrevistas, reforçando evidências das entrevistas para a definição categorias de análise.

5.1.1
Análise dos dados coletados – Questionários e Entrevistas

Como dito anteriormente, os dados coletados da fase de investigação do contexto de desenvolvimento de software foram compostos pelos questionários respondidos, as transcrições dos áudios das entrevistas e as anotações da pré-análise feita pela pesquisadora durante a transcrição dos áudios de entrevistas. A partir da consolidação e análise dos dados, foram definidas 5 (cinco) categorias de problemas ou questões identificadas durante a investigação do contexto de desenvolvimento de software, onde cada categoria apresenta subcategorias relacionadas. Para cada subcategoria, indicamos os fatores da tripla TNP relacionados às evidências desta subcategoria. Os fatores da TNP não são os mesmos para todas as subcategorias de uma dada categoria, mas há pelo menos um fator indicado em todas as subcategorias de uma dada categoria, como mostra a Tabela 5 que será detalhada a seguir.

Tabela 5. Categorias e subcategorias de evidências – Contexto de Desenvolvimento de software

Categoria Contexto de Desenvolvimento de software	Subcategoria	TNP
Construção de significado através do modelo	Conhecer UML *versus* Compreender UML	N-P
	Conteúdo mais importante que forma	N-P
Contexto do Processo de produção e consumo do modelo	Abandono de modelos	P
	Confiança nos modelos	P
	Modelos devem ser úteis	P
	Padrão de produção de modelos	P
	Produtores de modelos são consumidores exigentes de modelos	P
Ferramenta influenciando/impactando modelagem	Uso de desenhadores de modelos	T-N-P
	Validação UML por ferramenta de modelagem	T-N-P
Modelagem influenciada pelo contexto de desenvolvimento	Combinação de representações visuais e textuais	T-N-P
	Modelos práticos *versus* Modelos para documentar	T-N-P
	Nível de abstração de modelos	N-P
	Ponto ótimo do modelo	N-P
	Protocolo social para modelagem	T-N-P
	Vida útil de modelos	P
Modelos como objeto de coordenação	Objetos limítrofes do domínio como integradores	N-P
	Produção e/ou Consumo coletivo do modelo	N-P
	Representação do todo e das partes	N-P

Construção de significado através do modelo. Nesta categoria foram reunidos os problemas e questões identificados sobre a construção de significado através do modelo (Msem), considerando a notação UML (Mnot) como parte importante dos recursos para essa construção. Para esta categoria, temos os fatores notação (N) e pessoas (P) relacionados às evidências coletadas.

Contexto do Processo de produção e consumo do modelo. Nesta categoria temos os problemas e questões relacionados com algum ponto do processo de produção e consumo do modelo, destacando como tais problemas ou questões podem impactar o processo como um todo. Para esta categoria, temos somente o fator P (Pessoas) considerado, destacando características mais sociais que tecnológicas nas questões identificadas.

Ferramenta influenciando/impactando modelagem. Nesta categoria reunimos as questões relacionadas ao direto impacto da ferramenta utilizada, considerando suas características e comportamentos, na modelagem de software. Nesta categoria, temos os 3 fatores da tripla TNP considerados: Ferramenta (T), Notação (N) e Pessoas (P).

Modelagem influenciada pelo contexto de desenvolvimento. Nesta categoria foram reunidas as questões associadas às diretrizes de como o modelo (Mnot) será composto, qual será o seu propósito, se deve ser mantido ou descartado, combinações entre os envolvidos sobre o modelo, até que ponto o modelo deve ser detalhado e se é necessário o uso de ferramentas de modelagem. Esta categoria apresenta as decisões tomadas sobre o processo de produção e consumo de modelos antes que a modelagem seja realizada. Nesta categoria temos variações dos fatores da TNP entre as subcategorias, mas o fator pessoas (P) está presente em todas, já que as questões evidenciadas estão relacionadas ao contexto social onde o desenvolvimento de software é realizado.

Modelos como objeto de coordenação. Nesta categoria reunimos questões relacionadas aos objetos limítrofes (*boudary objects)* do domínio como integradores, que são conceitos que devem ser identificados e rastreados ao longo do processo de desenvolvimento; questões relacionadas à produção ou consumo coletivo do modelo por um grupo de pessoas e questões do modelo que pode representar o todo ou as partes de alguma visão do software, de acordo com a necessidade. Nesta categoria temos os fatores notação (N) e pessoas (P) considerados.

5.1.2
Achados– Questionários e Entrevistas

Os achados desta fase são descritos e evidenciados por trechos selecionados das entrevistas com os participantes, organizados e discutidos dentro de cada categoria e subcategoria.

Na categoria **Construção de significado através do modelo**, subcategoria ***Conhecer a UML versus Compreender a UML*** temos relatos de participantes sobre experiências onde o uso equivocado da notação (Mnot) acarreta problemas de comunicação através dos modelos (Msem). A notação (N) tem um papel

importante nessa comunicação entre pessoas (P), conforme se pode ver neste trecho:

"... as experiências ruins que eu tenho são exatamente pela falta de conhecimento da notação... aí os problemas aparecem..." (Participante 11)

Conhecer os elementos que compõem um modelo UML (Mnot) não é suficiente, é preciso saber como a combinação entre os elementos de um modelo deve ser feita e qual o seu significado (Msem). A representação gráfica oferece benefícios, porém esta deve expressar eficientemente o significado pretendido, ou então não serve seu propósito. (Petre, 1995) Aqui devemos considerar a comunicabilidade dos modelos de software como a capacidade crítica para tais artefatos. A comunicabilidade de modelos de software tem como base o conceito da comunicabilidade da Engenharia Semiótica, cuja definição é: a comunicabilidade pode ser definida como a capacidade do preposto do *designer* (interface de software) em atingir a metacomunicação completa, transmitindo aos usuários a essência da mensagem original do *designer*. (de Souza, 2005, p. 114) Esta metacomunicação será eficiente e eficaz se a comunicação é organizada e engenhosa (eficiente), e alcança o resultado desejado (eficaz). (de Souza et al., 2010) Estendendo esse conceito para os modelos de software, a representação do modelo (MNot) através da notação de modelagem, neste contexto a UML, deve ser capaz de transmitir aos consumidores do modelo a essência da mensagem original (Msem) do produtor do modelo, de forma eficiente e eficaz. A falta dessa comunicabilidade dos modelos influencia na adoção de tais artefatos no desenvolvimento de software:

"... as pessoas fazem modelos ruins, por isso elas acham que modelos não servem pra nada... aí elas falam: Não vou fazer modelo de caso de uso! É um monte de bolinha com uns bonequinhos, um ligado no outro. Não! Porque você faz de qualquer jeito! Coloca um monte de bolinha solta, com um monte de bonequinho solto e aí você acha que não serve pra nada! ... Aí você não lê!" (Participante 1)

Para que o produtor do modelo seja capaz de construir um modelo (Mnot) usando a notação UML, ele precisa compreender a notação, não somente conhecer. Deve ter a capacidade de interpretar (Msem) o que foi construído com a

UML (Mnot) para, só então, emitir o seu discurso utilizando essa linguagem específica. Essa precedência entre a capacidade de compreensão e capacidade de produção de discurso é uma característica cognitiva do ser humano, que é amplamente estudada no campo da aquisição (aprendizado) de linguagem natural, seja a primeira língua, no caso das crianças, seja numa língua adicional qualquer, no caso de crianças ou adultos. (Ingram, 1974) A comunicabilidade do modelo está diretamente relacionada com a capacidade do produtor do modelo em emitir um discurso organizado e engenhoso, que tenha potencial em alcança o resultado desejado.

Na subcategoria ***Conteúdo mais importante que forma***, temos relatos dos participantes sobre a importância dada aos problemas de modelagem, separando-os em problemas de conteúdo (Msem) e de forma (Mnot). Quanto à forma, uso da notação UML, os relatos indicaram que os problemas são considerados menos graves, já que quem consome o modelo conhece a UML e pode identificar os problemas e corrigi-los segundo sua interpretação. Nestes casos, segue notação (N) e pessoas (P) como fatores relevantes para apresentação e discussão das evidências:

"Já vi tantos tipos de modelos que hoje já consigo 'digerir' o que vier " (Participante 8)

Outro relato sobre a notação UML foi relacionado à importância do significado do artefato, seja usando a UML corretamente ou não. Num contexto onde várias pessoas estão fazendo a produção ou o consumo do modelo, os acordos durante esse processo podem ajudar na atribuição de significado (Msem) a um modelo (Mnot) mesmo que não esteja totalmente correto de acordo com a UML:

"se todas as pessoas concordassem o que aquilo ali ia significar, coloca assim mesmo, porque o importante era o entendimento... Deixa a UML pra lá!" (Participante 10)

Outro ponto relatado é a influência do conhecimento do domínio representado pelo modelo no processo de modelagem. Somente com o conhecimento do domínio, problemas relacionados ao conteúdo (Msem) expresso pelo modelo (Mnot) podem ser identificados. Porém, quem consome o modelo,

que tem algum conhecimento do domínio, pode optar em assumir um risco quando discorda de alguma parte do modelo (Msem) e segue com os seus entendimentos:

"eu conheço razoavelmente bem o domínio do problema e achei melhor começar a desenvolver alguma coisa, mesmo que esteja errado" (Participante 3)

O conhecimento do domínio relacionado ao conteúdo representado pelo modelo (Msem) de software pode ser um importante recurso para produtores e consumidores desses modelos. (Whittle et al., 2014) Porém, existe o risco do consumidor assumir certos entendimentos, mesmo que o modelo (Msem) esteja dando outras indicações, o que pode acarretar em retrabalho e problemas mais adiante no processo de desenvolvimento.

Esta situação está evidenciada por relatos na categoria **Contexto do Processo de Produção e Consumo do modelo,** subcategoria ***Abandono de modelos,*** onde o consumidor do modelo assume os riscos de ignorar a mensagem do modelo (Msem), e seguir com seus entendimentos. Neste caso temos o fator pessoas (P) em foco, onde o consumidor do modelo (Msem) decide ignorar a mensagem do produtor, elaborando sua própria mensagem. Temos vários P's interagindo nesse contexto, diretamente ou através do modelo:

"Isso aqui eu não entendi? Como eu conhecia o domínio então, eu imagino que seja assim, assim, assado. E aí terminei os modelos, comecei a transformar em código" (Participante 6)

Outras situações de abandono de modelos podem estar relacionadas ao modelo estar desatualizado e isso ser verificado por quem deseja consumir o modelo:

"eu vou tentando aproveitar aquilo. Agora se alguém mostrar, ou se a gente for comparar aquilo com o sistema e perceber que aquilo ali não representa a realidade, aí é a hora que você acaba abandonando... aí, quando você abandona um modelo e precisa daquele modelo depois, você prefere fazer um novo do que consertar um que tá melado..." (Participante 5)

Quando o consumidor do modelo identifica problemas de modelagem relacionados ao conteúdo (Msem) e à notação (Mnot) utilizada, o modelo pode ser

abandonado pelo consumidor e este passa a ser o produtor do modelo que precisa consumir:

"Putz! Deixa que eu faço!" (Participante 8)

Na subcategoria ***Confiança nos modelos*** temos relatos dos participantes sobre como a identificação de problemas de modelagem, sejam estes de conteúdo (Msem) ou forma (Mnot), pode impactar na confiança em relação ao modelo. O fator pessoas (P) segue como foco, na relação entre P's (P produtor e P consumidor):

"Geralmente quando eu começo a achar muitos problemas, eu paro de confiar totalmente no modelo e começo a tirar dúvida de tudo" (Participante 6)

"... os desenvolvedores normalmente não gostam (de modelos), não têm saco de ler, não confiam, normalmente eles estão desatualizados... em pouco tempo aquilo deixa de ser útil. " (Participante 5)

Houve um relado sobre a confiança no modelo estar associada ao seu produtor. Se uma determinada pessoa é definida dentro da equipe de desenvolvimento como alguém capaz de construir os modelos, esta pessoa tem a confiança da equipe:

"...se a pessoa tem autorização de alterar o modelo, a gente já confia que tem a base de UML e conhecimento do nosso processo. O que ela pode ter de deficiência é o entendimento do domínio do problema, ou mesmo do que nós entendemos como solução e aí você pode ter um desalinhamento do que eu acho que deve ser feito e o que ela acha. Aí a gente senta, alinha," (Participante 8)

Um relato nos indicou a relação entre problemas da subcategoria ***Confiança nos modelos*** com problemas da subcategoria ***Abandono do modelo*****:**

"... com certeza não tem a mesma confiança (em quem construiu o modelo) ... e dependendo desse nível de confiança, deixa esse modelo pra lá!" (Participante 11)

Na subcategoria ***Modelos devem ser úteis***, temos relatos dos participantes sobre a utilidade do modelo (Msem) em relação a sua corretude quanto ao uso da UML (Mnot), tendo como foco a comunicabilidade do modelo em relação aos envolvidos. Foco na comunicação entre pessoas (P):

"... os desenvolvedores não conheciam mais UML do que aquilo ali que a gente tava usando, entendeu? Era o suficiente pra gente se comunicar na verdade... não era tão crítico a gente checar as regras de modelagem em si..." (Participante 5)

Quais modelos e até onde estes devem ser detalhados têm relação direta com contexto do projeto de desenvolvimento. Idealmente essas decisões devem ser tomadas e combinadas entre todos os envolvidos. Tais decisões estão relacionadas à metodologia de desenvolvimento escolhida e a forma como o projeto será executado. (Pressman, 2010)

"Como no Scrum, a gente começa a fazer o mínimo entregável, então dali você vai evoluindo, então é uma evolução mais natural do que você desenhar tudo primeiro e depois voltar..." (Participante 9)

"Se não for considerar, melhor nem fazer o modelo. Em muitas empresas, o modelo é feito apenas para "constar", para passar em determinada inspeção." (Respondente 16)

"Embora sempre exista a expectativa de que o modelo seja utilizado, era conhecido que alguns modelos eram construídos por questões 'burocráticas'". (Respondente 7)

O propósito dos modelos deve estar claro para todos aqueles envolvidos no processo, sejam eles produtores ou consumidores do modelo, direta ou indiretamente. (Brambilla et al., 2012)

"... é fundamental para os participantes do projeto esse entendimento do que os modelos fazem pra equipe de desenvolvimento, eles são importantes e necessários, mas ao mesmo tempo entender que a comunicação deve ser mais natural e fluida possível sobre os modelos... " (Participante 11)

"Os modelos atrapalham quando ninguém os usa, por qualquer que seja o motivo." (Respondente 16)

Na subcategoria ***Padrão de produção de modelos*** temos relatos dos participantes sobre a identificação de padrões de produção do modelo (Mnot) durante o consumo dos modelos. Esses padrões são relacionados ao próprio produtor do modelo. Características e decisões que o produtor expressa no modelo

(Mnot) que são associadas ao estilo do produtor, a forma como este decidiu se comunicar com o consumidor do modelo:

"Eu não preciso que a pessoa siga um padrão específico sempre, mas desde que aquela pessoa siga o padrão dela sempre, entendeu? Uma coisa que ela fez no modelo A signifique a mesma coisa que ela fez no modelo B." (Participante 6)

"... o cara colocou uma ideia dele e esse cara das antigas, que coloca uma nomenclatura muito das antigas, num método muito antigo de fazer as coisas, os relacionamentos... "(Participante 12)

Na subcategoria ***Produtores de modelos são consumidores exigentes de modelos***, temos relatos dos participantes sobre a postura de produtores quando estão no papel de consumidores de modelos que têm experiência em produzir:

"Não tem jeito... mesmo lendo o modelo eu acabo olhando inconsistências, detalhes de modelagem... Não tem como colocar só cabeça de usuária... acaba ficando mais crítica..." (Participante 2)

Na categoria **Ferramenta influenciando/impactando modelagem**, temos relatos dos participantes nas subcategorias ***Uso de desenhadores de modelos*** e ***Validação da UML por ferramenta de modelagem.*** Nesta categoria, temos os 3 fatores relacionados, ferramenta (T), notação (N) e pessoas (P), nas evidências coletadas.

Na subcategoria ***Uso de desenhadores de modelos*** identificamos evidências de desenhadores serem considerados modeladores pelos respondentes e participantes. As ferramentas de modelagem apresentam problemas e limitações em relação à validação UML (Budgen et. al, 2011). O conhecimento de tais limitações e a ausência de expectativa de apoio em relação à validação UML, pode ser uma das razões pelas quais muitos profissionais respondentes (7/30) indicaram como ferramentas de modelagem, ferramentas que de fato são desenhadores de modelos (Cabot, 2011) em suas respostas sobre ferramentas conhecidas no questionário:

"Não tá servindo, vou pra uma coisa mais flexível... O modelo ele não gera nada, só usado pra discussão" (Participante 9)

“Visio eu acho que ele é prático, eu gosto, é mais pessoal, eu gosto muito da simplicidade pra criação de modelos...” (Participante 4)

“Já usei até Power Point!” (Participante 1)

Na subcategoria ***Validação da UML por ferramenta de modelagem*** temos evidências que indicam que produtores de modelos não têm expectativa que a ferramenta de modelagem seja capaz de garantir a conformidade em relação à notação UML. O conhecimento da notação UML é considerado com uma premissa pelos profissionais que produzem ou consomem modelos de software.

"Não, elas (ferramentas) não têm nenhum suporte pra isso... nenhum suporte em relação à notação, nenhuma inspeção, validação ou verificação do que você fez. Então era uma análise que gente fazia visual, enfim, usando a nossa experiência mesmo... um diagramador atendia melhor, não que atendia melhor, mas atendia bem, resolvia" (Participante 5)

“Eu não tenho essa expectativa em relação à ferramenta, mas eu sei, com certeza, que o Enterprise Architect, que já tem a modelagem certinha, com a notação correta, pra não permitir que você faça besteira, deve facilitar pra quem não tem o conhecimento de UML. No meu caso, como eu já trabalhei muito com isso, não preciso...” (Participante 4)

Na categoria **Modelagem influenciada pelo contexto de desenvolvimento,** destacamos relatos da subcategoria ***Combinação de representações visuais e textuais***, onde participantes indicaram a combinação de representações visuais e textuais para compor um artefato (Mnot) que efetivamente comunicasse o que o produtor do modelo (Msem) desejava. Nesta categoria temos os fatores ferramenta (T), notação (N) e pessoas (P) relacionados:

“Era só uma base comunicativa... era uma forma de aumentar o vocabulário, porque um fala uma coisa e outro outra, porque é ambíguo e tal, e aí a gente usava essa linguagem única, que era UML ou podia ser modelo E-R, não tinha só UML... era só uma forma, uma linguagem formal para que não houvesse ambiguidade, que todo mundo entendesse o que todos mundo tava falando” (Participante 10)

“Então nós temos em determinado momento as representações textualmente... essas especificações eram feitas via texto. Em algumas empresas

que eu trabalhei... eu participei de um trabalho que foi justamente pra gente tentar fazer uma representação em um modelo gráfico, em um modelo visual. Ou seja, utilizando de diagramas de atividades, nós tínhamos as raias, né? Então nós poderíamos utilizar como ator o sistema, as atividades eram os passos do texto, então nós conseguíamos representar o diálogo entre ator e sistema via diagrama de atividades, ou seja, modelo." (Participante 7)

"... Tá bom, a gente tá desenhando até aqui, a partir daqui a gente não vai representar graficamente... a gente vai representar em texto... enfim, vai descrever de alguma forma para não poluir demais..." (Participante 3)

Na categoria **Modelagem influenciada pelo contexto de desenvolvimento**, também destacamos relatos da subcategoria ***Modelos práticos versus Modelos para Documentar***, onde participantes relataram a diferença entre modelos práticos e modelos como documentação (Msem). Nesta categoria temos os fatores ferramenta (T), notação (N) e pessoas (P) relacionados:

"... a gente (empresa que desenvolve software para terceiros) quer entrar lá pra desenvolver software, no final das contas eu quero desenvolver software... e aí eu vou fazendo documentação..." (Participante 1)

"... como a gente já sabia que o projeto ia migrar, aí a gente acabou tomando mais cuidado (com os modelos como documentos)" (Participante 3)

Os modelos devem ser criados com um propósito, ou então nem devem ser criados (Ambler, 2002). No caso de modelos como documentação, essa decisão, de acordo com os relatos, está relacionada ao contexto do projeto de desenvolvimento de software. As decisões sobre que modelos serão produzidos, como serão usados e que ferramenta será utilizada influenciam na produção e no consumo dos modelos definidos para o contexto.

Na subcategoria ***Nível de abstração dos modelos*** foram identificadas evidências relacionadas ao nível de detalhamento dos modelos (Mnot) como sendo um fator que pode ajudar ou atrapalhar o consumo dos modelos (Msem). São evidências sobre a relação entre o nível de detalhamento de um modelo com o consumidor pretendido deste modelo e com o momento do desenvolvimento onde tal modelo é produzido ou consumido. Nestes casos, temos notação (N) e pessoas (P) como fatores predominantes na apresentação das evidências:

"Os modelos que são de mais detalhamentos são os mais consultados... Por ver os desenvolvedores mexendo no modelo de dados..." (Participante 1)

"Eram visões diferentes do mesmo conteúdo, mas não necessariamente era o mesmo modelo não (sobre evolução de modelo) ..." (Participante 2)

"Chegava um momento que você olhava pro modelo e ele deixava de ser representativo... o modelo ficava quase tão complexo quanto o código fonte... é possível sofrer com a 'overmodelagem'..." (Participante 3)

"... a preocupação era que o modelo 'compilasse', ou seja, gerasse código. Acredito que isso prejudicava bastante a legibilidade e consequentemente a manutenibilidade dos modelos." (Participante 3)

"Normalmente a gente acaba gerando mais de um pra poder atender aos diversos públicos porque já teve casos da gente precisar de um modelo mais técnico e aí o público leigo não consegui entender, teve casos da gente precisar de um modelo para esse público leigo, e aí não supria todos os detalhes que o pessoal técnico precisava." (Participante 5)

O nível de abstração do modelo (Mnot) é um fator que tem grande impacto na sua compreensão, na cognição das pessoas que consomem o modelo (Msem). (Stenning e Oberlander, 1995) Na tarefa de *design* de um software, são necessárias diferentes facetas, diferentes níveis de abstração para representar diferentes questões. (Myers et al., 2008) O nível de detalhamento está relacionado ao propósito do modelo, porém, no contexto de desenvolvimento de software, modelos são construídos por pessoas e pra que a construção seja possível, elas devem ser capazes de entender o que estão construindo, mesmo que o destino de tais modelos seja gerar código de software.

Na subcategoria ***Ponto ótimo do modelo***, identificamos relatos sobre a condição de parada da produção do modelo, de acordo com os objetivos intencionados com aquele modelo, quando um modelo é considerado 'finalizado' para seguir para a próxima fase do desenvolvimento. Nesta subcategoria temos os fatores notação (N) e pessoas (P) em destaque nas evidências coletadas:

"... foi uma decisão de projeto: "Eu não quero gerar todo o código a partir do modelo, vou gerar até uma parte, que isso vai permitir ter ali um ponto ótimo entre o modelo e o que ele gera..." não vou gerar 100%, mas em compensação

vou ter um modelo mais fácil, mais leve... então é você achar esse ponto de equilíbrio aí..." (Participante 3)

"Não tinha uma regra muito estabelecida não, era uma coisa de sentimento..." "Tá bom! Já temos aqui um modelo suficiente pra gente começar a desenvolver" (Participante 3)

As subcategorias ***Nível de abstração dos modelos*** e ***Ponto ótimo do modelo*** estão relacionadas, se considerarmos que uma vez definido o nível de abstração de um modelo (Msem), este deve ter um ponto ótimo de parada para que só represente (Mnot) o nível de abstração desejado pelo produtor. O esforço de ter um modelo absolutamente completo e consistente em relação à notação, não compensa considerando as propriedades do modelo. (Pressman, 2010) Definir esse ponto ótimo se apresenta como o desafio para que P se aproprie de N de tal forma que consiga construir os modelos (Mnot) desejados, alcançando os objetivos intencionados (Msem).

Na subcategoria ***Protocolo social para modelagem***, evidências relacionadas aos acordos sociais que são necessários para que os envolvidos no processo de modelagem alcancem uma convergência de entendimentos em relação ao modelo (Msem) produzido. Algumas evidências estão relacionadas à necessidade de extensão da notação utilizada, no caso UML, para que o modelo (Mnot) seja capaz de comunicar o que o produtor deseja. Nestes casos temos os 3 fatores, T-N-P, relacionados:

"... coloco uma legendinha e explico pra todo mundo o significado daquilo e pronto! Toca o barco!... normalmente pego um elemento da notação que a gente não usa, uso ele no significado que eu quero, e coloco uma legenda "(Participante 5)

"Aí quando eu pegar outra representação daquela pessoa... opa! Aquilo ali a gente já discutiu e combinou que ia representar desta forma" (Participante 7)

"se todas as pessoas concordassem o que aquilo ali ia significar, coloca assim mesmo, porque o importante era o entendimento" (Participante 10)

A necessidade de um protocolo social para ser usado de forma complementar ao protocolo tecnológico (Ellis e Gibbs, 1991) foi visto em várias evidências nas entrevistas e respostas de questionário. O protocolo social pode ser

usado para compensar alguma limitação da ferramenta, definição sobre o uso da notação ou mesmo uma formalização sobre a forma de trabalho das pessoas envolvidas. O protocolo social alinha entendimento dos envolvidos, procurando minimizar problemas durante um processo com fortes características de colaboração como no caso do desenvolvimento de software:

"...já tem a cultura aqui (uso de tags nos modelos)" (Participante 7)

"Gente! A gente comprou a ferramenta, mas não viu que não tinha isso aqui, então vamos fazer "assim-assado"? Beleza! Fazendo assim, a gente entende que é isso!" (Participante 12)

Na subcategoria ***Vida útil de modelos***, identificamos evidências relacionadas ao tempo que um modelo era efetivamente utilizado e, portanto, mantido durante o desenvolvimento. Nesta categoria temos somente o fator pessoas (P) relacionado, já que a vida útil dos modelos nos relatos está associada á forma como o desenvolvimento é executado, quantas pessoas estavam envolvidas, práticas de atualização do modelo e outras decisões no projeto de desenvolvimento de software:

"... o pró do MDA é essa facilidade de manter o modelo vivo, né? O modelo acompanhando o que está sendo desenvolvido de fato, né? Porque na experiência sem o MDA, a experiência que era desconexa, o grande desafio era esse: Garantir que a equipe tava mantendo o modelo atualizado pra no final ele refletir o que foi entregue." (Participante 3)

"...não tem muita formalidade, não tinha um documento passando de um lado pro outro, era o que surgia na hora e quando tinha muita gente conversando, a gente fazia no quadro branco. Se fosse pouca gente, fazia no papel mesmo... é o que eu chamo de papel de pão, que depois era jogado no lixo, não servia pra mais nada." (Participante 10)

Na categoria **Modelos como objetos de coordenação,** destacamos relatos sobre o papel dos modelos em apoiar a coordenação do trabalho das pessoas, orientando com uma linha conceitual do domínio relacionado ao software (subcategoria ***Objetos limítrofes do domínio como integradores***), representando a convergência do entendimento coletivo na produção e consumo do modelo (subcategoria ***Produção e/ou Consumo do modelo***) e como representação de

fronteiras no desenvolvimento (subcategoria ***Representação do todo e das partes***). Nesta categoria temos os fatores notação (N) e pessoas (P) em destaque nas evidências coletadas.

Na subcategoria ***Objetos limítrofes do domínio como integradores*** identificamos relatos dos participantes sobre a representação dos conceitos do domínio sendo representados pelos modelos (Mnot) e como estes conceitos devem ser identificados em diferentes fases do desenvolvimento de software:

"O que eu mais vejo eles mexendo é o modelo conceitual de dados e modelo lógico, porque tá lá todas as relações entre os dados, os conceitos que eles têm que manipular, nos objetos, nas classes. (Participante 1)

"... das classes, pra verificar se a gente não ter perdas, porque não adianta você fazer um bom trabalho em negócio e requisitos, e quando você começa a construir o software mesmo, fazer a tradução em objetos, se você tiver perdas, se você não tiver a mesma linguagem, procurar ter as mesmas associações, você vai construir um software que não está adequando, que não está aderente ao negócio..." (Participante 7)

Os objetos limítrofes (*boundary objects)* têm como objetivo viabilizar a comunicação e negociação entre comunidades diferentes que precisam colaborar para a realização de uma tarefa. (Lee, 2005) (Star, 2010) No contexto de desenvolvimento de software, os conceitos principais do domínio devem sempre estar representados nos artefatos, portanto produtores e consumidores devem ter essa preocupação, já que os objetos limítrofes são uma forma de fazer a rastreabilidade entre um modelo (Msem) e outro, garantindo que conceitos centrais do domínio estão sendo representados e tratados nos artefatos associados ao software desenvolvido.

Na subcategoria ***Produção e/ou Consumo coletivo do modelo*** identificamos evidências nos relatos dos participantes sobre a coautoria na produção do modelo de todos que participam da discussão sobre o modelo (Mnot), o que apresenta um comprometimento coletivo sobre o que foi representado no modelo (Msem). Também identificamos evidências sobre o consumo coletivo do modelo usado em discussões, onde todos colaboram e constroem os seus próprios entendimentos a partir do modelo representado e do que os demais externalizam.

"... como é que eu vou estruturar essa informação, aí a gente acaba sentando junto e projetando mesmo, aí sai um modelo." (Participante 9)

"... a gente sempre fazia o modelo no quadro, num quadro branco e uma pessoa ia lá desenhava uma coisa, aí outra pessoa podia ir lá e comentar em cima "acho que isso tá errado e tal", pegava a caneta e reformar o modelo..." (Participante 10)

"Quando a gente vai documentar esse modelo já houve muita discussão... o pessoal de negócio, o pessoal de requisitos, eventualmente pessoas mais experientes do time de construção, pra definir essa estrutura, modularização, a arquitetura como um todo." (Participante 8)

"Em relação a entendimento, é uma decisão tomada por todos, quando todos entenderam (o que o modelo representa), ok." (Participante 1)

Na subcategoria ***Representação do todo e das partes*** identificamos evidências nos relatos dos participantes sobre o modelo (Mnot) ser usado para representar relações e fronteiras no desenvolvimento:

"Eu consigo modelar os primeiros, porque eu sei qual o conjunto de funcionalidades iniciais que eu vou desenvolver e aí tenta fazer, não um esquema de desenvolvimento ágil nem tanto, mas acaba fechando pequenos pacotes pra poder ter coisa funcionando." (Participante 1)

"... precisa delimitar escopo pro projeto... existem áreas tipo supervisão e existem áreas operacionais. Todas as áreas operacionais almejam trazer melhoras para os seus processos, só que nem sempre o patrocinador que, por estratégia ou falta de grana, ele vem e fala: "Ah! Eu vou priorizar isso aqui porque não tenho grana pra bancar o projeto inteiro"." (Participante 7)

A representação através dos modelos de software pode ter diferentes níveis de abstração para diferentes propósitos, como evidenciado subcategoria ***Nível de abstração dos modelos*** da categoria **Modelagem influenciada pelo contexto**. Quando o modelo (Mnot) representa um escopo de desenvolvimento, este pode ser utilizado para tomar decisões em relação às partes desse escopo, como, por exemplo, que parte será desenvolvida primeiro ou quem irá desenvolver qual parte. O modelo é o que Prates (1998) caracteriza como objeto de coordenação. Através deste objeto os envolvidos conseguem colaborar para a construção do

software, de forma que cada um conhece a sua parte e pode, assim, contribuir para a construção do todo.

Após a investigação do contexto de desenvolvimento de software, decidimos aumentar nosso escopo e realizar a investigação da interação com ferramentas de modelagem de software, considerando como cenários a produção e o consumo de modelos. As evidências nos mostraram que a visão mais ampla do processo de modelagem de software (atividades de produção e consumo de modelos) seria um contexto mais completo e enriquecido de questões relacionadas ao processo como um todo e a atividades particulares de partes desse processo.

5.2
Investigação da Interação com Ferramentas de Modelagem

Na fase de investigação da interação com de ferramentas de modelagem de software, procuramos inspecionar, avaliar e caracterizar a emissão da mensagem dos *designer*s de ferramentas de modelagem de software. Selecionamos 4 (quatro) ferramentas de modelagem de software, considerando seu reconhecimento junto à indústria de desenvolvimento de software. Para isso, escolhemos duas referências: o *Magic Quadrant for Enterprise Architecture Tools do Gartner* e a UML Vendor Directory Listing indicada pela OMG (Object Management Group).

O *Magic Quadrant for Enterprise Architecture Tools do Gartner*[1] é uma referência sobre as ferramentas mais relevantes disponíveis no mercado para diversos fins, inclusive para apoio ao desenvolvimento de software. A *Gartner* é uma empresa de pesquisa e consultoria que definiu e usa a metodologia do *Magic Quadrant* para realizar uma análise qualitativa de um determinado mercado e suas direções, maturidade e participantes. No *Magic Quadrant* as ferramentas são categorizadas como *leaders, challengers, visionaries e niche players*, conforme mostrado na Figura 18. Neste quadrante identificamos os fornecedores de 2 (duas) das ferramentas selecionadas para esta pesquisa: IBM, da ferramenta IBM Rational Software Architect, e Sparx Systems, da ferramenta Enterprise Architect.

[1] http://www.gartner.com/technology/research/methodologies/research_mq.jsp

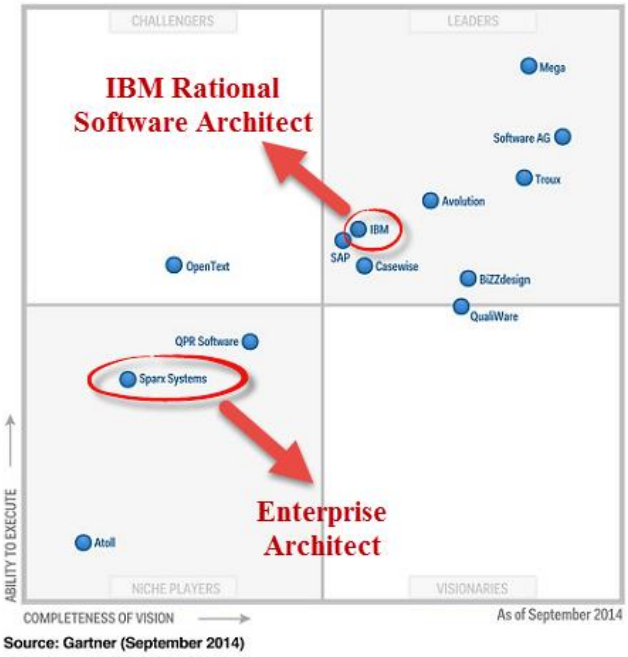


Figura 18. Magic Quadrant for Enterprise Architecture Tools 2014

Na *UML Vendor Directory Listing*[2] estão as ferramentas de modelagem de software recomendadas pela OMG (*Object Management Group*), responsável pela especificação da UML (Figura 19). Nesta lista, temos 3 (três) das ferramentas selecionadas para esta pesquisa: Astah, Enterprise Architect e Visual Paradigm.

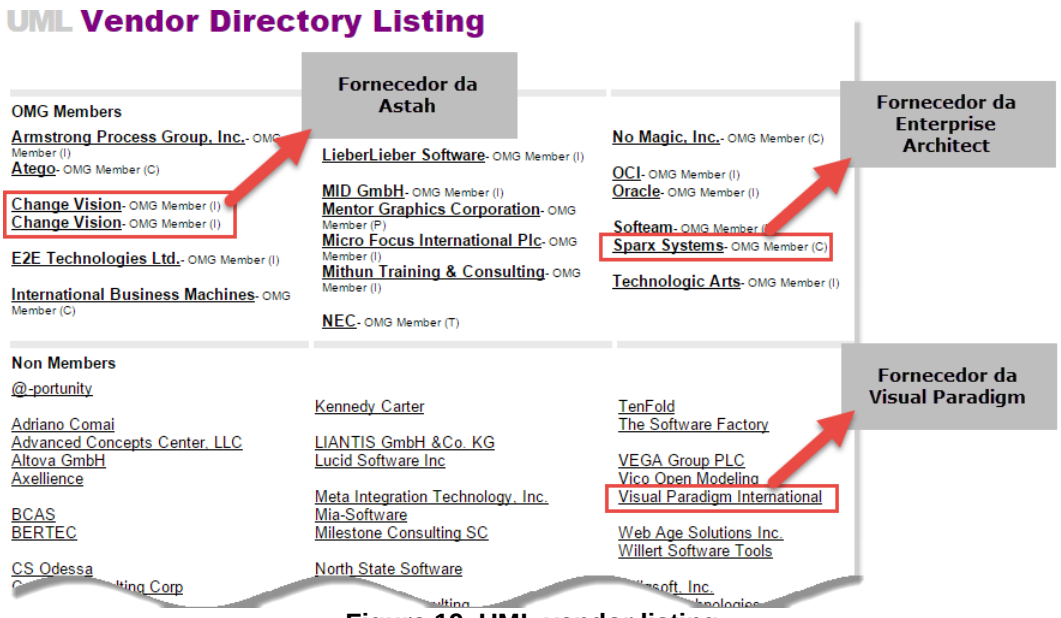


Figura 19. UML vendor listing

[2] http://uml-directory.omg.org/vendor/list.htm

As versões selecionadas para a avaliação foram as versões mais completas das ferramentas, disponibilizadas como *trial* nas páginas dos fornecedores[3]:

- Astah Professional 6.9
  http://astah.net/editions/professional
- Visual Paradigm Enterprise Edition 12.0
  http://www.visual-paradigm.com/editions/enterprise.jsp
- Enterprise Architect Ultimate 11
  http://www.sparxsystems.com.au/products/ea/index.html
- IBM Rational Software Architect 9.0.0.1
  http://www.ibm.com/developerworks/downloads/r/architect/index.html

Após selecionarmos as ferramentas definimos os cenários de avaliação a serem considerados para a avaliação. A partir do que coletamos e observamos na etapa de investigação do contexto de desenvolvimento de software, identificamos duas perspectivas para a avaliação: perspectiva do produtor de modelos de software e perspectiva do consumidor de modelos de software. As duas perspectivas se complementam e nos apresentam um contexto real do contexto de desenvolvimento de software, onde modelos são produzidos e consumidos pelas pessoas envolvidas no processo.

Durante a etapa de entrevistas com os profissionais de desenvolvimento de software na fase de investigação do contexto de desenvolvimento de software, observamos que a produção e o consumo do modelo estão fortemente ligados. Um modelo é construído considerando um objetivo de uso, um consumidor pretendido. Portanto, o produtor deve ter uma visão, mesmo que não completamente definida, do que será feito com o modelo de quem irá consumir o modelo para que ele seja capaz de produzir o modelo de uma maneira mais informada, considerando este contexto mais amplo do que somente a atividade de produção do modelo usando uma notação e uma ferramenta de modelagem. Além disso, incluindo o cenário de consumo de modelos, temos um novo espaço de possibilidades de interação com a ferramenta, que pode ser avaliado tanto de

[3] As versões das ferramentas selecionadas para o estudo desta tese estavam disponíveis para *trial* em Fevereiro de 2015.

forma mais pontual quanto considerando o processo de produção e consumo. A combinação dos dois cenários possibilitou a investigação da relação entre a produção e consumo de modelos.

Os cenários estão relacionados a um modelo de atividades que, hipoteticamente, será parte do processo de desenvolvimento de um *Sistema de Recursos Humanos (SRH).* O domínio do modelo de atividades em si é o processo de *Requisição de Férias*. No cenário de produção, o produtor do modelo tem um conjunto de informações sobre o processo de *Requisição de Férias*, a partir dos quais deve construir o modelo de atividades. O produtor também está ciente de que tal modelo será utilizado por outra pessoa no processo de desenvolvimento para a identificação das regras de negócio associadas ao SRH em construção. Os dois cenários, de produção e consumo de modelos de software, utilizados na avaliação das ferramentas selecionadas estão apresentados na íntegra no Apêndice D.

Para a realização das avaliações das ferramentas estabelecemos algumas premissas para assegurar que o foco ficasse no suporte que a ferramenta oferece para a execução das metas inerentes ao cenário de inspeção. As premissas estabelecidas foram:

Modelos são partes de um projeto na ferramenta. Consideramos que o produtor irá compartilhar o modelo (Mnot) construído com o consumidor visado através dos recursos de disponibilização de projeto para colaboração, oferecidos pela ferramenta. Ou seja, o consumidor do modelo também vai utilizar a ferramenta para executar o seu cenário. Determinamos essa forma de disponibilização como condição de contorno para todas as avaliações, procurando uma uniformidade na execução dos cenários em todas as quatro ferramentas analisadas. Cada ferramenta oferece um conjunto particular de formas de compartilhamento de modelos, mas em todas elas os modelos (Mnot) podem ser acessados através de um projeto de modelagem criado na ferramenta.

Forma de representação das regras de negócio. Identificamos nas entrevistas com os profissionais que a necessidade de representação de regras de negócio em modelos (Mnot) é uma necessidade real e muitas vezes problemática no desenvolvimento de software:

"... se eu quero colocar regras de negócio e eu não tenho como coloca ***regra de negócio****, eu uso nota. Aí coloco uma legendinha e explico pra todo mundo o significado daquilo e pronto! Toca o barco!" (Participante 5)*

"Por que foi pedido? Porque existia uma regra de negócio. Então naquele momento também tinha que ser feita uma referência à ***regra de negócio****, então depende muito da ferramenta." (Participante 7)*

A representação de regra de negócio em um modelo (Mnot) pode ser feita de diversas formas, mas para a avaliação das ferramentas nesta pesquisa, decidimos representar as regras de negócio utilizando elemento 'nota' (*note* ou *comment*) presente em todas as ferramentas avaliadas, associada ao recurso de estereótipo definido pela própria UML que também é suportado por todas as ferramentas avaliadas. O estereótipo (*stereotypes*) é um mecanismo de extensão da UML, junto com *tags* e *constraints*. Um estereótipo representa a variação de um elemento existente do modelo (Mnot) com a mesma forma (assim como atributos e associações), mas com uma intenção diferente. (OMG, 2015)

5.2.1
Análise dos dados coletados – Avaliação de Ferramentas

Os dados coletados na fase de investigação do contexto de desenvolvimento de software foram coletados durante a avaliação das ferramentas selecionadas, utilizando o CSCE associado à tripla TNP. Realizamos a avaliação de 4 (quatro) ferramentas de modelagem de software, definidas como tal por seus fornecedores, e reconhecidas no mercado de desenvolvimento de software (Cabot, 2015b). Tínhamos por objetivo gerar uma coleção de dados representativos sobre a ferramenta, escolhida como o foco desta fase da pesquisa.

A análise dos dados foi feita considerando as evidências identificadas nas avaliações de todas as ferramentas avaliadas. A partir desta análise, definimos 5 (cinco) categorias de questões ou problemas observados durante a avaliação, com subcategorias mais específicas. Para cada subcategoria, relacionamos os fatores da

tripla TNP que compõem as evidências daquela subcategoria. Novamente, os fatores da TNP não são os mesmos para todas as subcategorias de cada categoria, mas pelo menos um deles é indicado em todas as subcategorias de uma categoria. As categorias, e suas respectivas subcategorias com os fatores da TNP relacionados, estão apresentadas na Tabela 6.

Tabela 6. Categorias e subcategorias de evidências – Investigação da Interação com ferramentas de modelagem

Categorias de Investigação de interação com ferramentas de modelagem	Subcategorias	P	C	TNP
Problemas com recursos para estender/adicionar funcionalidades	Suporte para criação de *plug-ins*	**X**	**X**	**T-P**
	Disponibilização de *plug-ins* criados por outros desenvolvedores, usuários da ferramenta	**X**	**X**	**T-P**
Problemas com recursos para construção de significado baseado em múltiplas notações	Combinação de notação UML com outras notações	**X**	**X**	**T-N-P**
	Identificação visual das regras de negócio		**X**	**T-N-P**
	Recursos de desenho no modelo	**X**		**T-P**
	Recursos para alterar representação visual de elementos	**X**		**T-N-P**
	Suporte para localização do usuário no modelo	**X**	**X**	**T-N-P**
Problemas com recursos para construção de significados baseado somente em UML	Algumas restrições da UML são implementadas	**X**		**T-N-P**
	Conhecimento sólido da UML e seus elementos	**X**	**X**	**T-N-P**
Problemas no suporte ao processo de desenvolvimento de software	Consumidor pode fazer o mesmo que o produtor		**X**	**T-P**
	Diversas opções de *menu* para exportar e importar de/para diversos formatos	**X**	**X**	**T-P**
	Mais do que um modelador, mas principalmente um modelador	**X**	**X**	**T-N**
	Modelo sempre em construção	**X**		**T-P**
	Suporte à colaboração e trabalho em equipe	**X**	**X**	**T-P**
Problemas na interação com modelos utilizando, principalmente, o *mouse*	Produção de modelos utilizando, principalmente, o *mouse*	**X**		**T-P**
	Consumo de modelos utilizando, principalmente, o *mouse*		**X**	**T-P**

P – Categoria identificada no cenário de Produção de modelo
C - Categoria identificada no cenário de Consumo de modelo
TNP – Fatores da tripla TNP relacionados à subcategoria

Como indicando na Tabela 6, algumas subcategorias foram identificadas nos dois cenários avaliados: Produção e Consumo de modelos. Apesar das avaliações terem sido executadas considerando cada um dos cenários separadamente, a análise foi realizada considerando o processo de produção e consumo de modelos de forma mais abrangente. A partir desta análise, as categorias foram identificadas e definidas conforme descritas abaixo:

Problemas com recursos para estender/adicionar funcionalidades. Esta categoria reúne as evidências relacionadas aos recursos que as ferramentas apresentam para que o usuário possa estender as funcionalidades da ferramenta e, portanto, fazer algo mais do que o *designer* definiu quando construiu a ferramenta. Nas avaliações realizadas identificamos duas situações que foram separadas em subcategorias: o usuário da ferramenta tem suporte para criar novos *plug-ins*, e o usuário tem disponível uma coleção de *plug-ins* criados por outros usuários. A possibilidade de criação de *plug-ins* para estender funcionalidades da ferramenta foi identificada em todas as ferramentas avaliadas, enquanto a disponibilização de plug-ins criados por outros usuários foi identificada em 3 das 4 ferramentas avaliadas.

Problemas com recursos para construção de significado baseado em múltiplas notações. Nesta categoria reunimos as evidências relacionadas aos recursos disponíveis na ferramenta para que o usuário possa construir significado através do modelo (Msem), seja através da produção do modelo ou no momento do consumo. As evidências estão relacionadas à possibilidade de combinação da notação UML com outras notações, aos recursos de identificação visual de elementos específicos, no caso do cenário, as regras de negócio, aos recursos de desenho no modelo (Mnot), aos recursos para alterar representação visual de elementos e ao suporte para localização do usuário no modelo (Mnot).

Problemas com recursos para construção de significados baseado somente em UML. Nesta categoria reunimos as evidências relacionadas aos recursos disponíveis na ferramenta para que o usuário possa construir significado

utilizando a notação UML. As evidências estão relacionadas aos recursos disponíveis na ferramenta relacionados à notação UML, seus modelos (Mnot), elementos, restrições e recursos de extensão, particularmente para o uso de estereótipos.

Problemas no suporte ao processo de desenvolvimento de software. Nesta categoria reunimos as evidências que indicam ao usuário que a modelagem é parte de um processo, e apresenta recursos para que o modelo (Mnot) possa ser usado como ponto de partida desse processo. As evidências estão relacionadas ao consumidor poder realizar as mesmas ações de interface que o produtor do modelo, disponibilidade de diversas opções de *menu* para exportar e importar de/para diversos formatos, características focadas na modelagem em si, recursos que indicam que o modelo está sempre em construção e funcionalidade que oferecem suporte a colaboração e trabalho em equipe.

Problemas na interação com modelos utilizando, principalmente, o mouse. Nesta categoria reunimos as evidências que indicam que a interação para execução das atividades de produção e consumo de modelos é feita principalmente com o uso do mouse. Esta é uma categoria mais relacionada à usabilidade das ferramentas para apoiar as ações de 'manipulação' dos modelos durante a interação do usuário com a ferramenta.

5.2.2
Achados – Avaliação de Ferramentas

Os achados das avaliações das ferramentas resultaram de uma análise que considerou o contexto do processo de modelagem de software, tendo em conta os dois cenários de inspeção. Portanto, devido ao foco mais abrangente da avaliação, alguns pontos que consideramos menos relevantes para a discussão da comunicabilidade e usabilidade das ferramentas não foram destacados como evidências. Um exemplo seria um problema de usabilidade relativo ao dimensionamento da representação gráfica das atividades no Astah que é feito de acordo com o tamanho do nome do elemento. Resulta disso que cada elemento aparece com um tamanho diferente. Se o usuário desejar, ele pode redimensionar manualmente cada representação, ou pode selecionar todas e utilizar a funcionalidade Alignment>Ajust size para redimensionar várias de uma vez, ou

ainda definir o mesmo tamanho de todos os elementos selecionados. No Apêndice E apresentamos, como exemplo, um dos diagramas de atividades criados no cenário de produção executada na ferramenta Astah.

Na categoria **Problemas com recursos para estender/adicionar funcionalidades** destacamos evidências relacionadas à possibilidade de extensão das funcionalidades da ferramenta de modelagem através de *plug-ins*. Todas as ferramentas avaliadas apresentam recursos e guias de como desenvolver *plug-ins*, evidências da subcategoria ***Suporte para criação de plug-ins das ferramentas***, como exemplificamos nas Figura 20 e Figura 21. Na maior parte dos casos, essa mensagem é passada pelo *designer* através de signos metalinguísticos (conteúdo de help, website), mas no caso do Astah, por exemplo, temos um signo estático (item de *menu*) indicando esta capacidade de extensão. Também foram identificadas evidências. Na subcategoria ***Disponibilização de plug-ins criados por outros desenvolvedores, usuários da ferramenta*** identificamos evidências para duas das ferramentas avaliadas: Astah (Figura 20) e Enterprise Architect (Figura 22).

Figura 20. *Plug-in* como recurso de extensão da ferramenta – Astah

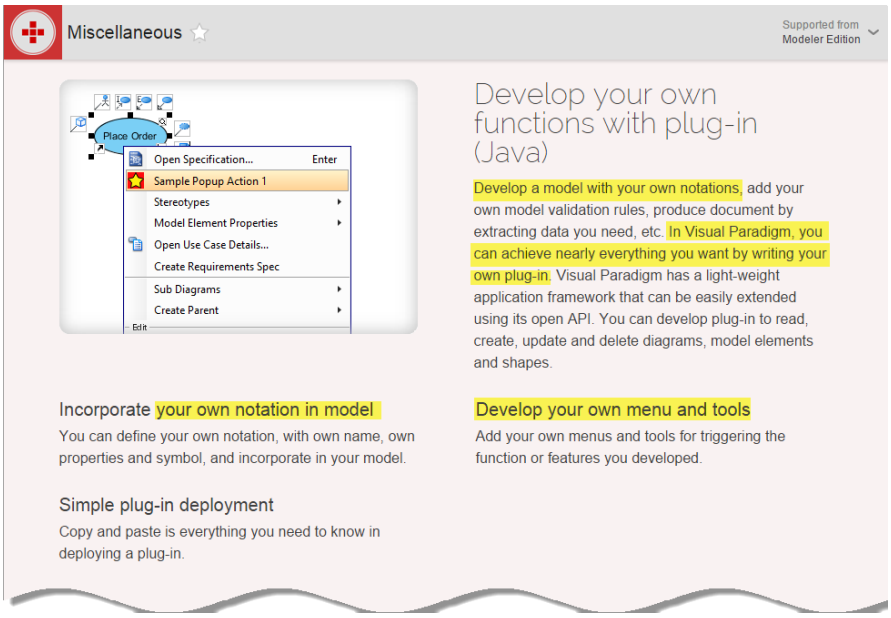


Figura 21. *Plug-in* - Visual Paradigm

Figura 22. P*lug-ins* desenvolvidos por parceiros - Enterprise Architect

A possibilidade de estender as funcionalidades da ferramenta através do uso e, principalmente, do desenvolvimento de *plug-ins*, evidencia que aquele sistema notacional que o *designer* da ferramenta idealizou, oferece recursos para que o usuário utilize as notações oferecidas de outra forma ou mesmo crie novas. O *designer* da ferramenta dá recursos para que o usuário expanda o sistema de notações, e consequentemente de significados, como desejar. Esta característica foi associada à CDN **Notação secundária**, que é a habilidade de se usar as notações diferentes da sintaxe formal oferecida para expressar informação ou

significado. No caso da Visual Paradigm (Figura 21), por exemplo, o *designer* destaca que com o desenvolvimento de um *plug-in*, o usuário também tem a possibilidade de estender a própria notação de modelagem (Mnot), criando a sua própria.

Estas evidências foram relacionadas aos fatores ferramenta (T) e pessoas (P) da tripla. T oferece recursos para que P possa expandir as possibilidades de funcionalidade e interação durante o uso de T. T oferece a P recursos para criar novas notações dentro do contexto da ferramenta, expandir o sistema de significados idealizado originalmente pelo *designer* da ferramenta.

Na categoria **Problemas com recursos para construção de significado baseado em múltiplas notações**, destacamos evidências de 3 (três) subcategorias, que estão fortemente relacionadas: ***Combinação de notações UML com outras notações, Recursos para alterar representação visual*** e ***Recursos de desenho no modelo***.

1) Na subcategoria ***Combinação de notações UML com outras notações*** destacamos evidências relacionadas à possibilidade de combinar diferentes representações e notações (*i.e.* Diagrama de Entidade-Relacionamento, Tabela de requisitos, BPMN, Mapa mental, *Sketching*, etc) com modelos UML em um mesmo projeto (Figura 23). Apresentando essas possibilidades de combinação de representações, através de signos metalinguísticos e estáticos, o *designer* indica que entende que o seu usuário precisa ter opções de representação para escolher a que melhor se aplica num determinado projeto ou momento do desenvolvimento.

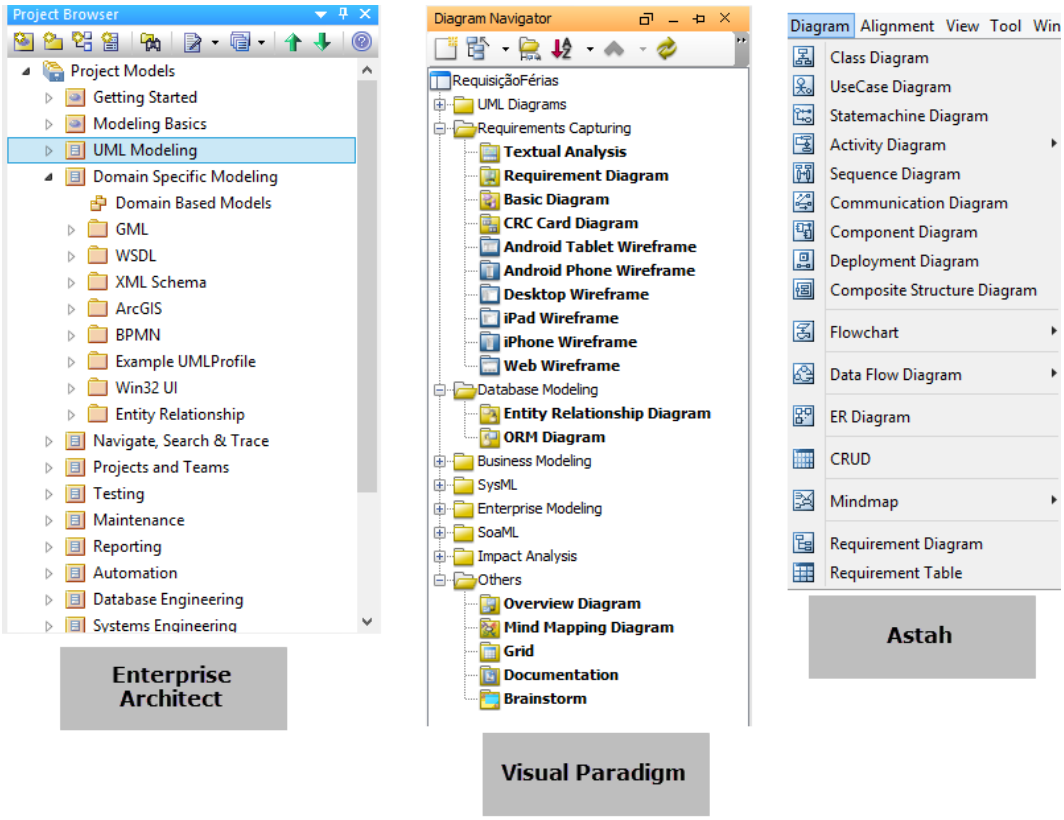


Figura 23. Combinação de diversos modelos e representações UML

Esta característica foi associada a duas dimensões cognitivas: **Visibilidade,** os diagramas UML são apresentados juntos com as demais representações oferecidas em um mesmo item do *menu*, visualmente relacionados, indicando a possibilidade de combinação de diferentes notações em um mesmo projeto da ferramenta, e **Proximidade de mapeamento**, já que a possibilidade de combinar notações oferece flexibilidade similar ao que desenvolvedores de software têm quando constroem modelos em um quadro branco, por exemplo. Ter diferentes recursos de representação durante a construção de um modelo (Mnot) é interessante para a atividade de modelagem em si, com ou sem ferramenta de apoio.

Essas evidências foram relacionadas aos fatores ferramenta (T), notação (N) e pessoas (P), já que as ferramentas oferecem recursos para que o usuário construa o modelo combinando outras notações com a UML, dando mais recursos para o usuário construir significados necessários e desejados.

2) Na subcategoria ***Identificação visual das regras de negócio*** temos evidências na atividade de consumo do modelo, quando a ferramenta oferece recursos para o usuário visualizar o modelo (Mnot) como um todo, enquanto é possível

ver os detalhes de uma determinada parte na área de construção do modelo (Figura 24).

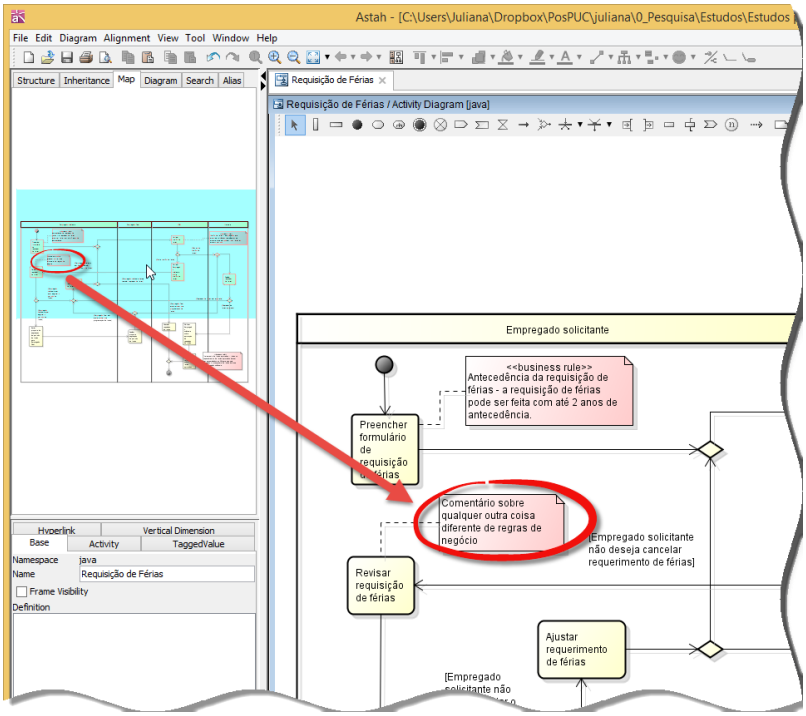


Figura 24. Visão "*Map*" da Astah

As evidências desta subcategoria foram associadas à dimensão cognitiva **Visibilidade**, já que a ferramenta provê diferentes formas e recursos para facilitar a visualização do modelo (Mnot), o que ajuda o usuário quando há a necessidade de diferentes visões (de parte ou do todo) do modelo (Mnot).

Essas evidências foram relacionadas aos fatores ferramenta (T), notação (N) e pessoas (P), já que a ferramentas oferece recursos para que o usuário tenha visões diferentes do modelo, porém a visualização da notação através da ferramenta pode ser impactada pelas decisões tomadas durante a produção do modelo (Mnot). Por exemplo, se um mesmo elemento da UML for usado com diferentes propósitos, isso pode não ficar claro na visão geral, conforme evidenciado pela visão *Map* do Astah na Figura 24. Visualmente pode parecer o mesmo elemento na visão geral do modelo (Mnot), porém ter significados diferentes quando investigado no detalhe, impactando assim o consumo do modelo (Msem).

3) Na subcategoria ***Recursos de desenho no modelo*** destacamos evidências relacionadas aos recursos de desenho disponíveis junto com as notações do modelo (Figura 25 e Figura 26). Recursos como inclusão de formas geométricas, inclusão de texto, inclusão de imagem, desenho livre e destaque,

são disponibilizados para serem usados no modelo (Mnot), porém sem uma sintaxe definida. São recursos visuais que podem ser utilizados para representar alguma necessidade de significação do modelo (Msem).

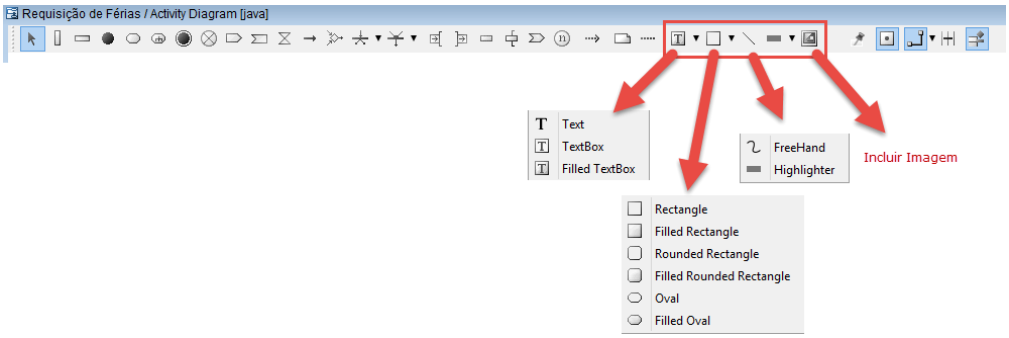


Figura 25. Recursos para desenhar no modelo – Astah

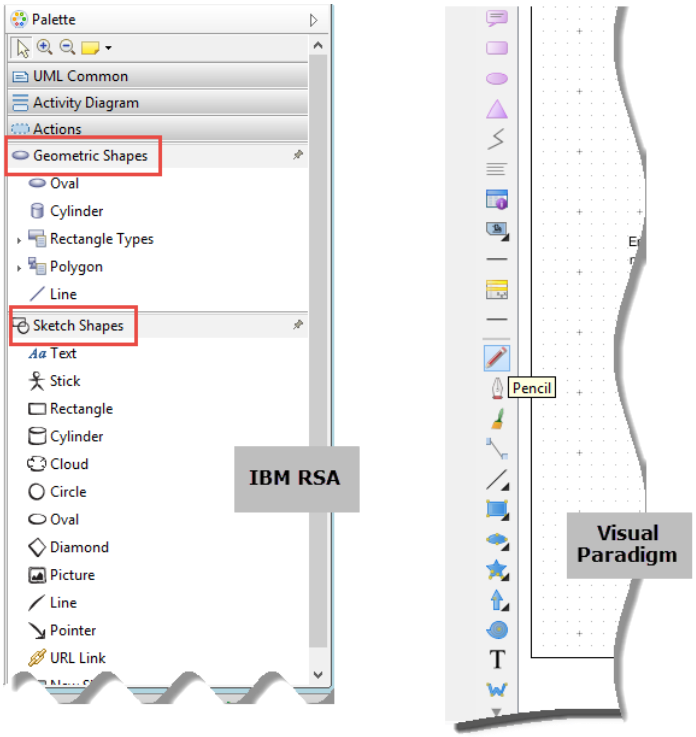


Figura 26. "Desenhar" no modelo - IBM RSA e Visual Paradigm

Associamos essa evidência à dimensão cognitiva **Provisoriedade**, onde existe flexibilidade da notação da ferramenta para que os usuários possam brincar com ideias sobre a representação do modelo (Mnot). Oferecendo recursos de desenho, o *designer* está possibilitando ao usuário experimentar ideias enquanto constrói o modelo (Mnot), ou mesmo como recursos para colaboração entre pessoas na construção do modelo como exemplificado na Figura 27. Esse tipo de notação dá flexibilidade ao usuário enquanto interage com o modelo.

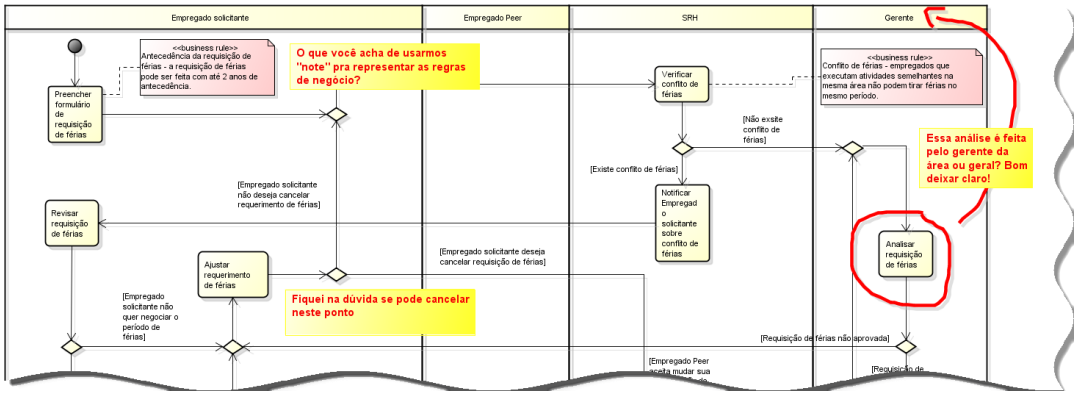


Figura 27. Brincando e colaborando com o modelo - Astah

Essas evidências foram relacionadas aos fatores ferramenta (T) e pessoas (P), já que a ferramenta oferece ao usuário a possibilidade de explorar diferentes significados no modelo através do uso de elementos de desenho até que uma representação final do modelo seja definida.

4) Na subcategoria ***Recursos para alterar representação visual de elementos*** destacamos evidências relacionadas à representação visual dos modelos e seus elementos. Quando um elemento do modelo está selecionado, algumas opções da barra de ferramentas se tornam disponíveis. São opções relacionadas à representação visual dos elementos, como recursos de mudança de cor de linha, preenchimento, fonte e inclusão de um ícone no elemento (Figura 28 e Figura 29). Esses recursos apresentam a possibilidade do construtor do modelo (Mnot) poder criar novas representações visuais dos elementos, porém sem alterar o seu comportamento.

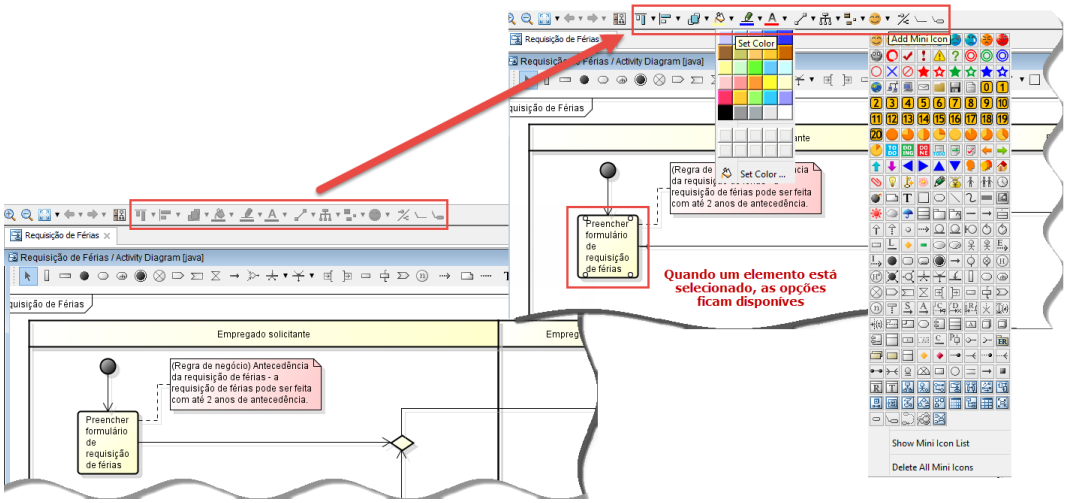


Figura 28. Recursos de alteração de representação visual – Astah

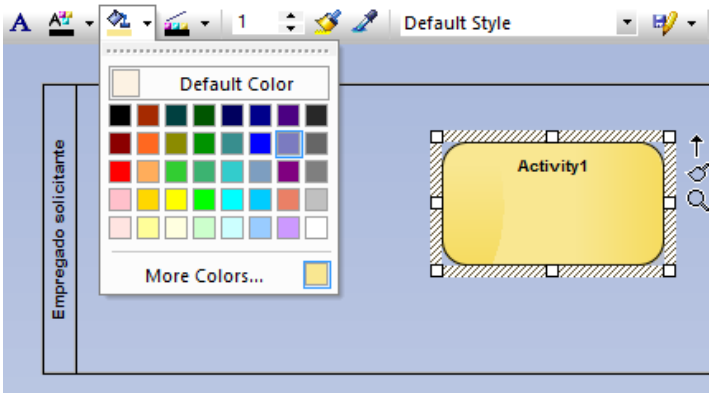


Figura 29. Alteração de representação visual – Enterprise Architect

Esta característica foi associada a duas dimensões cognitivas: **Notação secundária** e **Dependências ocultas.** Quando a ferramenta oferece diversos recursos para que o usuário possa alterar a representação visual de elementos, alterando cor de preenchimento, de linha, incluindo ícone, criando novas representações visuais com o uso de formas geométricas e outros, dá flexibilidade para se criar novos significados a serem apresentados nos modelos construídos. Temos mais uma vez a dimensão cognitiva **Notação secundária**.

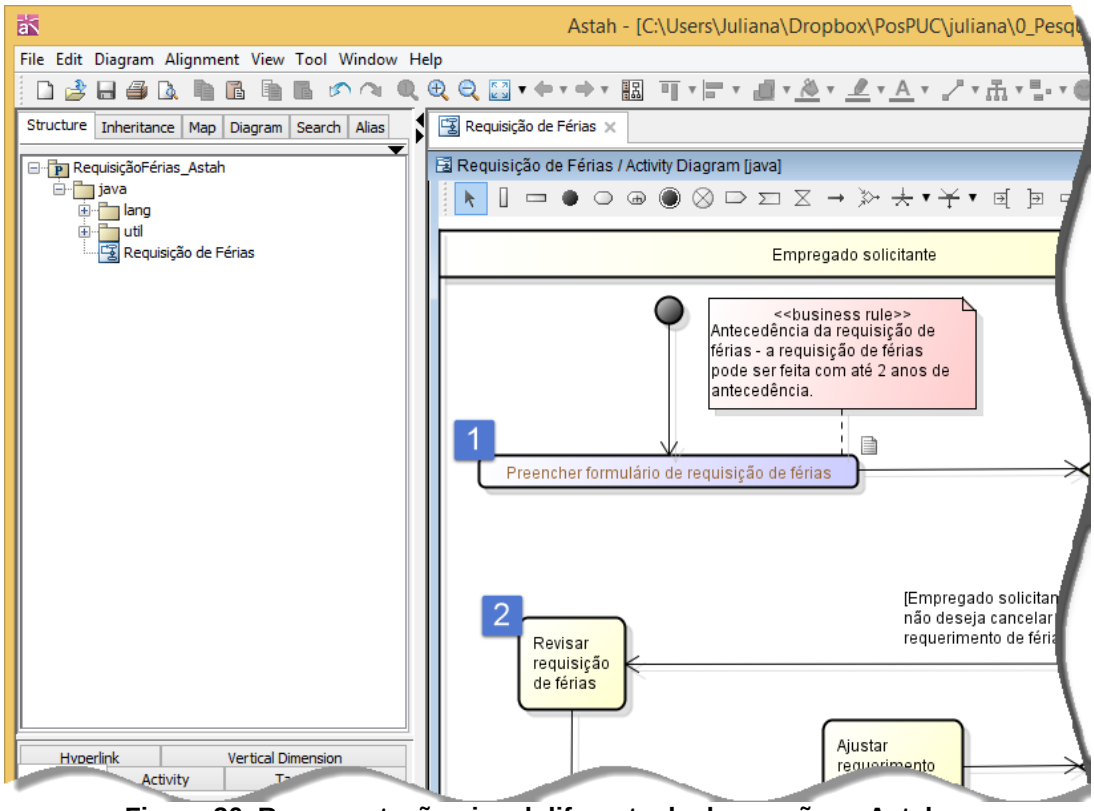


Figura 30. Representação visual diferente de duas ações - Astah

Também associamos esta evidência à dimensão cognitiva **Dependências ocultas**, identificada quando existe uma relação entre duas entidades, mas essa

relação não é totalmente visível. Os recursos para alterar e criar representações visuais podem ser bastante úteis, porém quando um elemento da UML tem a sua representação visual alterada, suas propriedades permanecem as mesmas. Na Figura 30, os elementos indicados com 1 e 2 são elementos do tipo *action* do diagrama de atividades da UML. O usuário só consegue ter ciência dessa situação, se comandar explicitamente a comparação entre as propriedades dos elementos, selecionando um de cada vez e indo propriedade a propriedade para ter certeza. Não é oferecida nem uma dica para que o usuário possa rapidamente verificar se dois elementos são do mesmo tipo. Essa característica é similar em todas as ferramentas avaliadas.

Essas evidências foram relacionadas aos fatores ferramenta (T), notação (N) e pessoas (P), já que a ferramenta oferece ao usuário a possibilidade de atribuir novos significados aos elementos da própria UML, ampliando assim seus recursos de expressão quanto ao modelo (Msem) que está sendo construído.

5) Na subcategoria ***Suporte para localização do usuário no modelo*** destacamos evidências relacionadas aos recursos que a ferramenta apresenta para que o usuário tenha conhecimento sobre qual é o executor cada atividade está relacionada, no caso do diagrama de atividades. Tanto para a produção quanto para o consumo do modelo, essa relação entre atividade e executor é bastante relevante para o entendimento do diagrama de atividades. Normalmente, quando o diagrama é maior do que fica legível na tela, é necessário utilizar as barras de rolagem vertical e horizontal para navegar pelo diagrama. Das ferramentas avaliadas, duas oferecem recursos para que o executor da raia esteja sempre visível ao usuário em qualquer ponto do diagrama (Figura 31 e Figura 32).

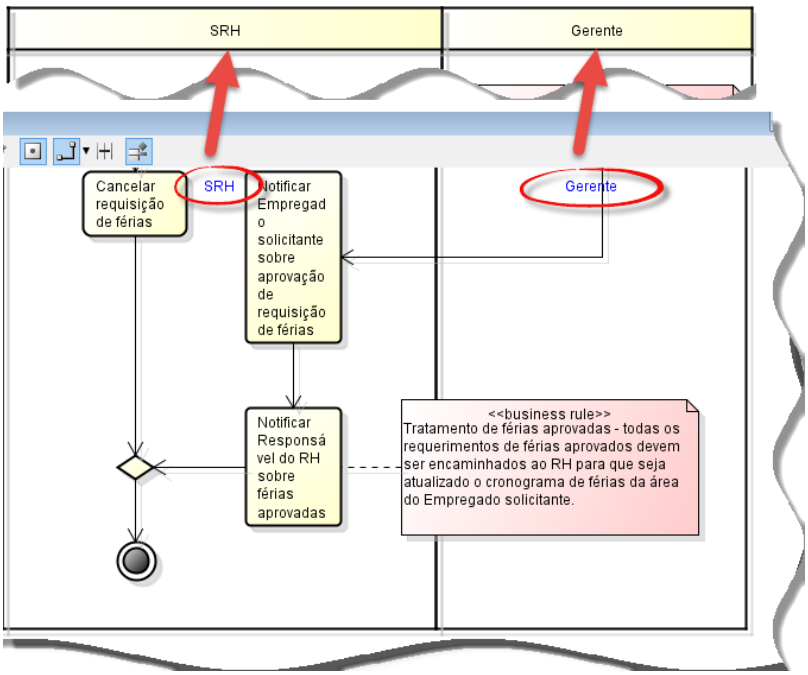


Figura 31. Nome do executor da raia sempre visível - Astah

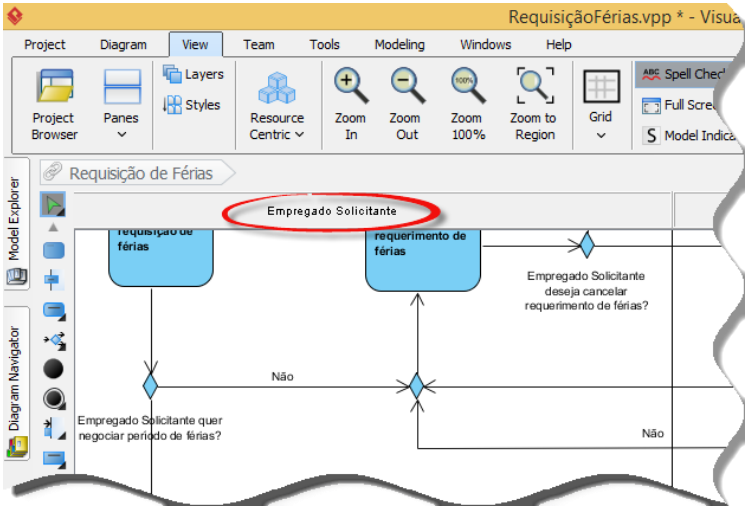


Figura 32. Nome do executor na raia sempre visível - Visual Paradigm

Essas evidências foram relacionadas às dimensões cognitivas **Visibilidade e Expressão de papéis**. Na dimensão **Visibilidade**, temos a característica de que o executor da atividade, que é representado pelo nome dado ao elemento *partition*, ou raia, da UML, está sempre visível ao usuário. Mesmo quando o topo da raia não está na tela, o nome do executor fica visível ao usuário. Na dimensão **Expressão de papéis** temos explicita a relação entre o executor e as atividades contidas em uma raia em qualquer ponto do modelo. O usuário ser capaz fazer essa relação no contexto do diagrama de atividades é importante para o consumo do modelo (Msem).

Essas evidências foram relacionadas aos fatores ferramenta (T), notação (N) e pessoas (P), já que a ferramenta (T) oferece ao usuário (P) a possibilidade de estar ciente a qualquer momento da relação executor-atividade, definida pela UML (N), durante a produção ou consumo do modelo (Msem) de atividades.

Na categoria **Problemas com recursos para construção de significados baseado somente em UML** temos duas subcategorias: ***Algumas restrições da UML são implementadas*** e ***Conhecimento sólido da UML e seus elementos.*** As evidências desta categoria foram relacionadas aos fatores ferramenta (T), notação (N) e pessoas (P). A ferramenta (T) oferece ao usuário (P) alguns recursos para apoiar o uso da UML (N).

1) Na subcategoria ***Algumas restrições da UML são implementadas*** destacamos evidências relacionadas às restrições da UML que são implementadas nas ferramentas. Algumas dão alertas durante a construção do modelo (Figura 33) e outras nem oferecem recursos para ação que não pode ser realizada. Um exemplo é o caso do evento final que não apresenta seta de transição de saída como os demais elementos, já que um evento final não pode ter transição de saída (Figura 34).

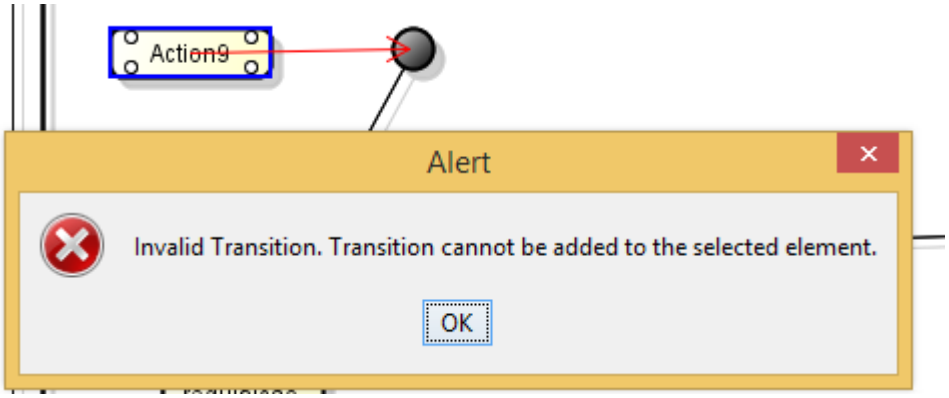


Figura 33. Alerta após usuário tentar realizar uma transição– Astah

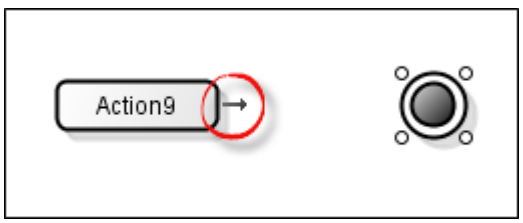


Figura 34. Para o evento final não tem transição de saída

Além dos recursos durante a construção do modelo apresentados nas evidências acima, foi identificado em três das ferramentas um recurso de validação da UML. No caso do IBM RSA, essa validação pode ser executada a qualquer momento através da seleção de um item de *menu* (Figura 35). Porém não

fica claro para o usuário até onde vai essa validação. Por exemplo, se o modelo tiver elementos soltos, sem transição (situação de modelagem em geral), nenhuma mensagem é apresentada; da mesma forma se o diagrama de atividade não tiver um estado final (restrição da UML), também não há nenhuma indicação ao usuário.

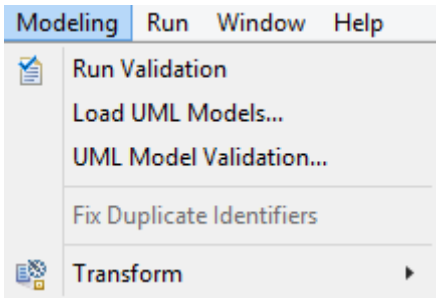
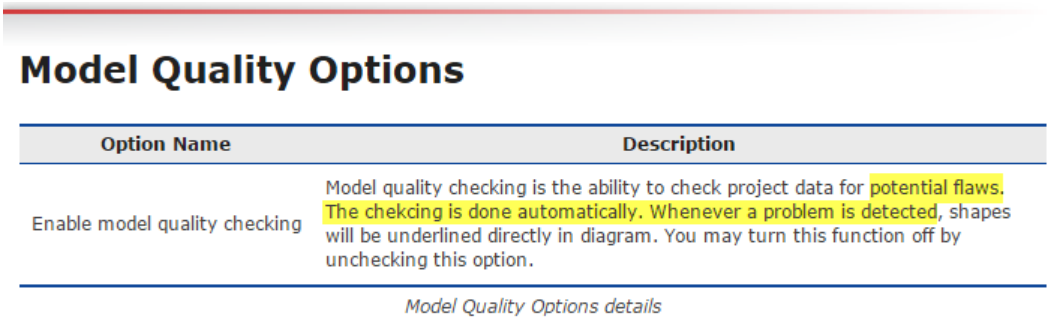


Figura 35. UML validation - IBM RSA

No caso do Visual Paradigm, o usuário tem a opção de habilitar ou não a funcionalidade chamada *Model Quality*, que segundo conteúdo de ajuda da ferramenta (Figura 36) verifica falhas em potencial, mas não deixa claro que falhas são essas, se estão relacionadas a restrições da notação de modelagem ou não.



Model Quality Options

Option Name	Description
Enable model quality checking	Model quality checking is the ability to check project data for potential flaws. The chekcing is done automatically. Whenever a problem is detected, shapes will be underlined directly in diagram. You may turn this function off by unchecking this option.

Model Quality Options details

Figura 36. Model Quality (Help content) - Visual Paradigm

Essa opção de verificação da qualidade do modelo é desabilitada por *default*, ou seja, deixa o usuário à vontade para fazer essa verificação quando e se quiser. Porém, quando o usuário habilita a função, a verificação não é executada prontamente. Em algum momento, que não é do conhecimento do usuário, essa verificação será feita e os problemas serão destacados no modelo (Figura 37).

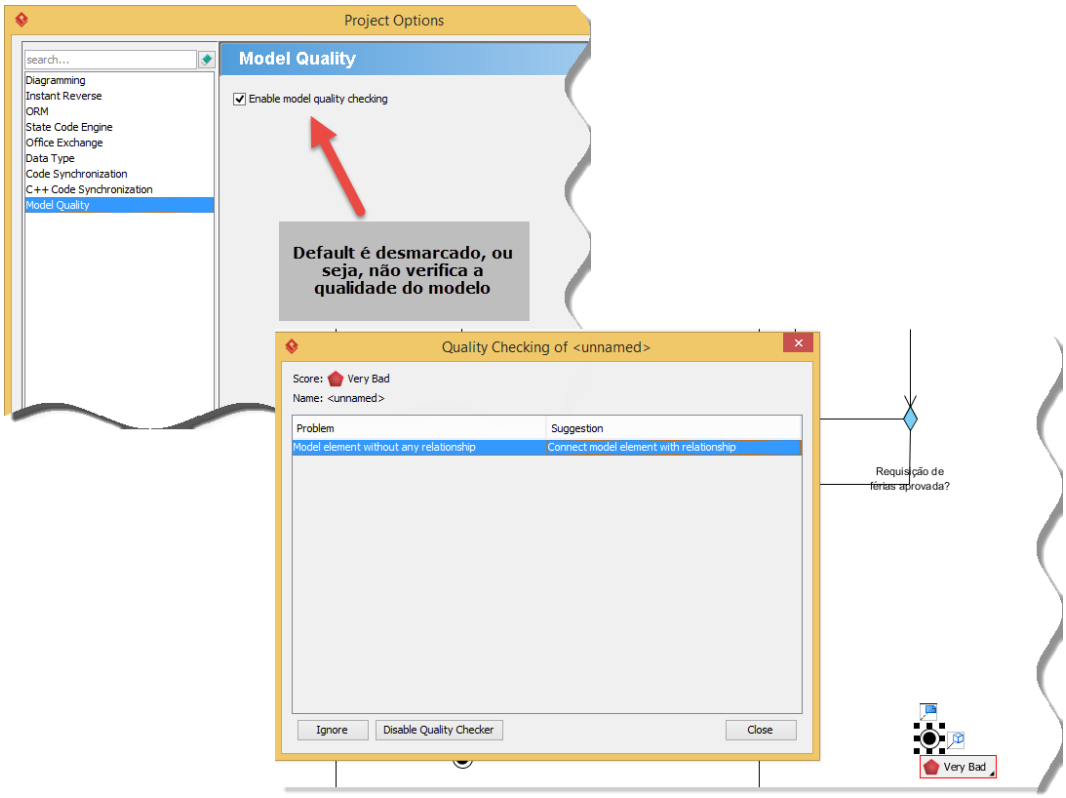


Figura 37. *Model Quality* (em execução) - Visual Paradigm

Essas evidências foram associadas à dimensão cognitiva **Propensão a erros**, que está relacionada à tendência da notação para conduzir a erros contra os quais o sistema dá pouca proteção. O usuário tem a opção de validar ou verificar a qualidade de seu modelo UML, porém o alcance e o momento de realização dessa verificação não ficam claros para o usuário e em alguns casos pode induzir o usuário a pensar que não existem problemas com o seu modelo, quando na verdade existem.

2) Na subcategoria ***Conhecimento sólido da UML e seus elementos,*** destacamos evidências sobre a disponibilização dos elementos do diagrama de atividades da UML. Na avaliação de todas as ferramentas, os elementos são listados apresentando sua representação gráfica e, em alguns casos, o nome do elemento (Figura 38). Quando e qual elemento utilizar é uma decisão do usuário que, portanto, deve ter um bom conhecimento da notação UML do diagrama de atividades.

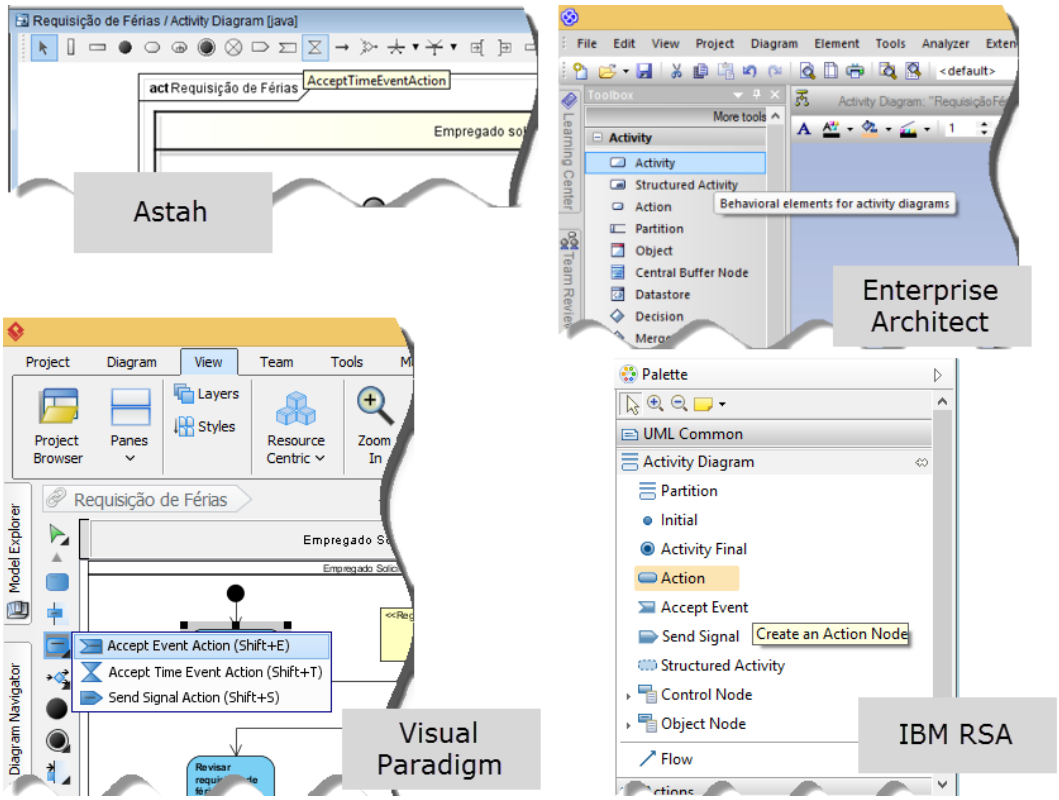


Figura 38. Elementos da UML - Diagrama de atividades

Relacionamos essas evidências à dimensão cognitiva **Operações mentais difíceis**, onde há uma alta demanda de recursos cognitivos para que os usuários entendam a notação e acompanhem o que está acontecendo. A notação UML é extensa e conta com diversos elementos. Mesmo que o usuário possua um conhecimento sólido da notação, a ferramenta poderia ajudá-lo a fazer a relação de forma mais fácil dos elementos disponíveis com o modelo (Mnot) como um todo. No caso do Enterprise Architect, por exemplo, a mesma mensagem é apresentada na dica de todos os elementos (*Behavior elements for activity diagram*) quando o usuário seleciona o elemento com o *mouse*.

Na categoria **Suporte ao processo de desenvolvimento** destacamos evidências das seguintes subcategorias: ***Consumidor pode fazer o mesmo que o produtor, Diversas opções de menu para exportar e importar de/para diversos formatos*** e ***Modelo sempre em construção*.**

1) Na subcategoria ***Consumidor pode fazer o mesmo que o produtor*** destacamos evidências relacionadas ao uso da ferramenta pelo consumidor do modelo, tendo acesso os mesmos recursos de edição que o produto do modelo, o que

pode atrapalhar o consumidor do modelo durante a interação com o modelo (Figura 39).

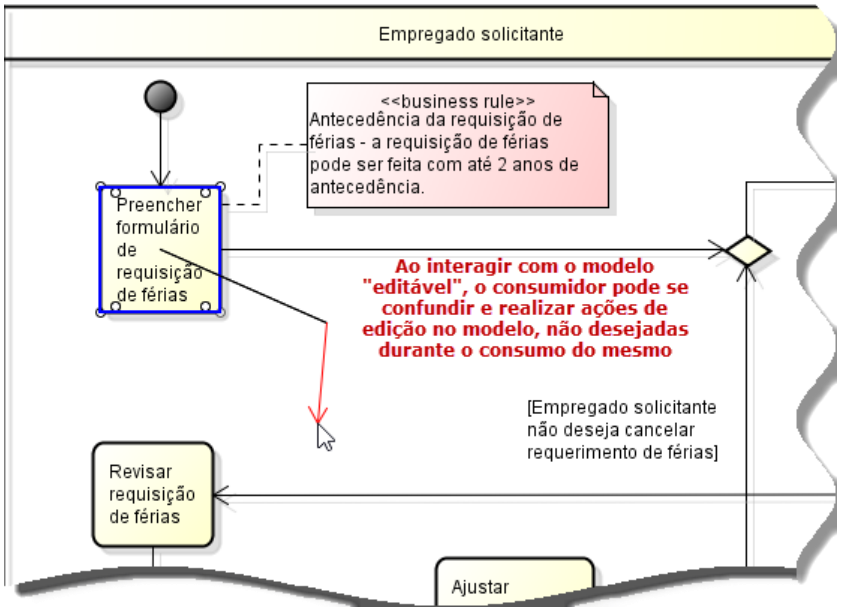


Figura 39. Edição durante consumo de modelo

No caso do Astah, e também do Visual Paradigm, existe o conceito de '*Reference Model'* quando um projeto pode ter uma cópia criada como *read-only*. (Figura 40) Esta cópia criada mantém a referência com o modelo original, e pode ser atualizada conforme o modelo de referência é alterado (ação que deve ser executada pelo usuário). O projeto *read-only* tem um signo ao lado dos modelos (Figura 41), além de notificar ao usuário quando este editar o modelo, indicando que o mesmo é somente de referência (Figura 42).

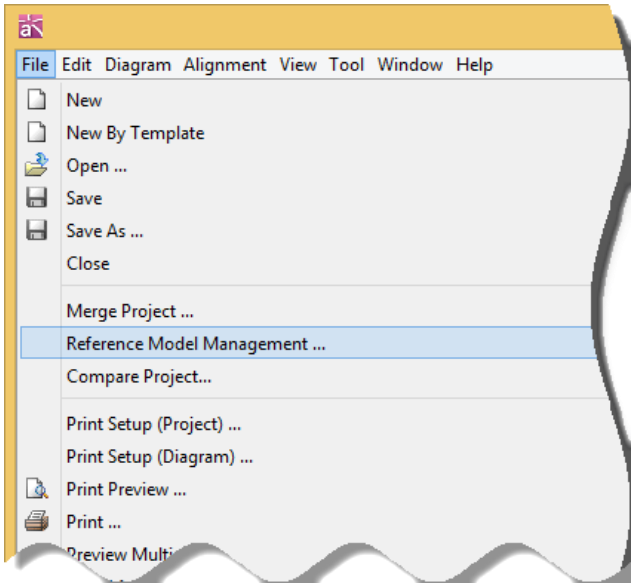


Figura 40. Opção de criação de modelo de referência - *read-only*

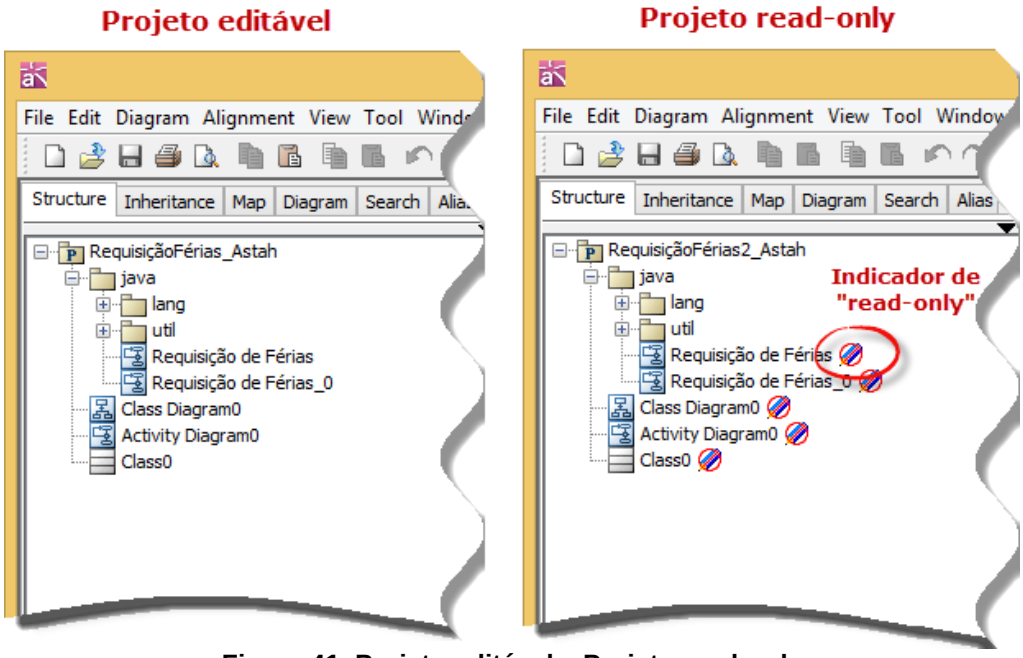


Figura 41. Projeto editável e Projeto read-only

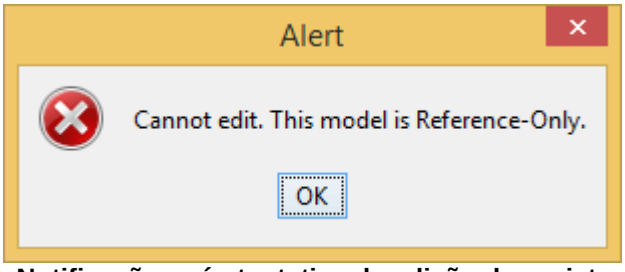


Figura 42. Notificação após tentativa de edição de projeto *read-only*

Porém, o uso dessa funcionalidade somente é feito se o usuário assim desejar. Como o compartilhamento do projeto, onde estão os modelos, pode ser feito simplesmente por arquivo (compartilhar o arquivo *. 'extensão da ferramenta' com outras pessoas), a decisão de usar essa opção de um projeto *read-only* está fora do contexto da ferramenta e pode dizer muito sobre a relação entre produtor e consumidor. Se o produtor compartilhar um projeto como *read-only*, pode indicar alguma desconfiança do produtor na capacidade do consumidor em relação ao uso "livre" do modelo. Outra possibilidade é o consumidor gerar esse modelo *read-only* (se ele tem o modelo editável, pode executar a ação), o que indicaria uma insegurança dele em interagir com um modelo editável. De qualquer forma o uso dessa funcionalidade, nos diz muito sobre o contexto social do uso da ferramenta.

O produtor do modelo, ao compartilhar um modelo *read-only*, comunica ao consumidor do modelo que ele entende que este consumidor é alguém que não

tem capacidade de interagir com um modelo editável. Neste ponto, o produtor está especificando o discurso do *designer*:

“Eis a minha visão de quem você, ... ”

***Designer*:** *“Você pode ou não ter a capacidade de consumir um modelo editável.”*

Produtor: *“Você não tem capacidade de consumir um modelo editável”*

Relacionamos essas evidências à dimensão cognitiva **Operações mentais difíceis**. O consumidor do modelo, tendo as mesmas funcionalidades do produtor disponíveis, pode se colocar, mesmo que sem querer, em situações de produção do modelo (ex. puxar uma nova transição a partir de uma atividade, deslocar um elemento e etc.) sendo necessária uma ação de desfazer, o que acaba atrapalhando a atividade de consumo em si. A notação acaba dificultando o usuário na atividade.

Essas evidências foram relacionadas aos fatores ferramenta (T) e pessoas (P), já que a decisão de disponibilizar ou não um modelo editável, utilizando algum recurso de T, ao consumidor do modelo (P) vai depender do produtor do modelo (P). O produto (P) faz algum juízo sobre o consumidor (P) para tomar essa decisão.

2) Na subcategoria ***Diversas opções de menu para exportar e importar de/para diversos formatos*** destacamos evidências relacionadas às funcionalidades que indicam que o modelo é um artefato dentro de um processo, que depois da modelagem, outras atividades são realizadas, oferecendo formas de compartilhar os modelos, como em arquivos de imagem, csv e pdf, e transformar este modelo em código-fonte de diversas linguagens, base de dados e etc. (Figura 43 e Figura 44).

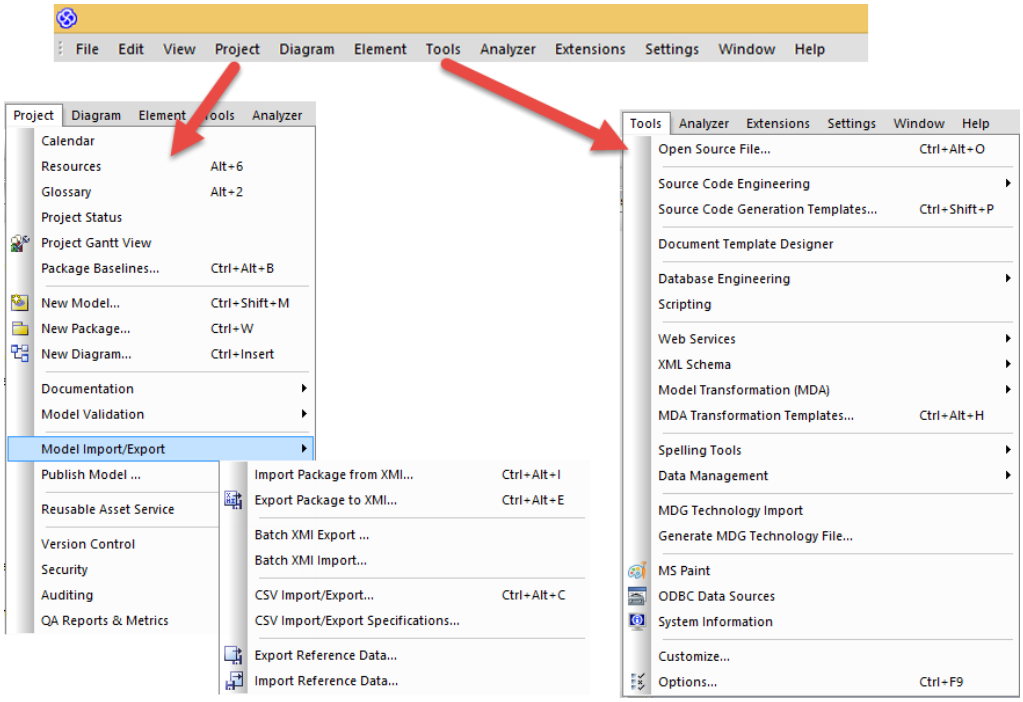


Figura 43. Opções importação/exportação - Enterprise Architect

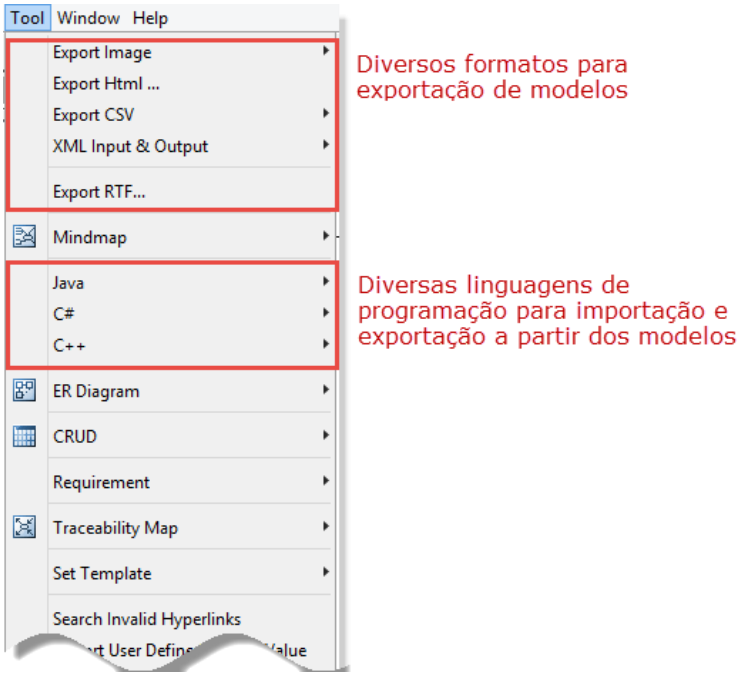


Figura 44. Opções importação/exportação - Astah

Essas evidências foram relacionadas às dimensões cognitivas **Abstração** e **Expressão de papéis.** A ferramenta destaca seu objetivo principal de apoiar a modelagem UML, oferecendo outras notações e representações que podem ser combinadas, porém também oferece funcionalidades de compartilhamento e transformação dos modelos (Mnot) construídos em outros artefatos (código, banco de dados, etc.). Portanto, temos o modelo em níveis de abstrações diferentes:

como um artefato a ser construído e como um artefato relacionado a outros artefatos no processo de desenvolvimento, características da dimensão **Abstração**. Também podemos associar essa relação do modelo com outros artefatos, indicando pelas possibilidades de compartilhamento e transformação, com a dimensão **Expressão de papéis,** que se define quando o propósito de uma entidade e como esta entidade se relaciona com todo o componente são óbvios e pode ser inferido diretamente.

Essas evidências foram relacionadas aos fatores ferramenta (T), notação (N) e pessoas (P), já que a ferramenta (T) dá suporte à pessoa (P) para construir modelos usando (N). Ao mesmo tempo dá suporte ao compartilhamento e transformação dessas modelos (Mnot), colocando o modelo dentro do contexto de desenvolvimento de software, onde outras pessoas (P) normalmente estão envolvidas.

3) Na subcategoria ***Mais do que um modelador, mas principalmente um modelador*** identificamos evidências que destacam o objetivo principal das ferramentas que é a modelagem de software. Na descrição das ferramentas e funcionalidades apresentadas o foco fica sempre na atividade de modelagem (Figura 45 e Figura 46). Todas as outras funcionalidades disponíveis (exportação, geração de código e etc.) só são possíveis a partir ou para criar um modelo (Mnot) elaborado na ferramenta.

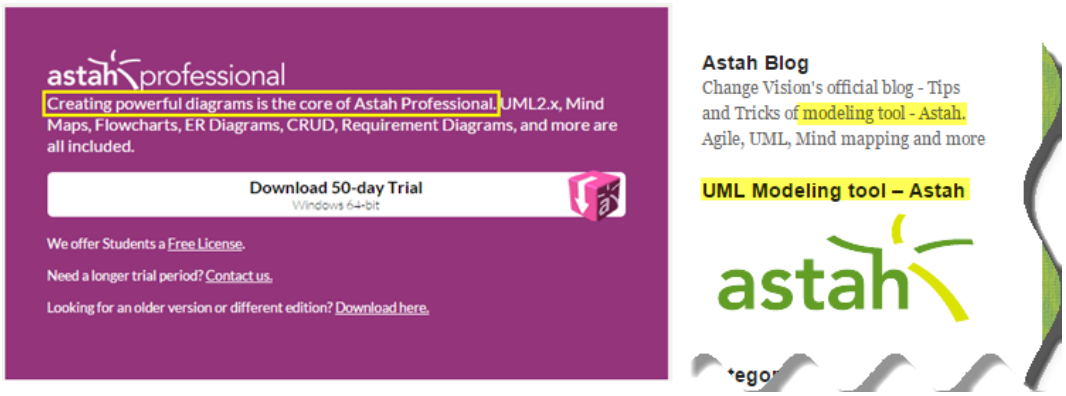


Figura 45. Foco na modelagem - Astah

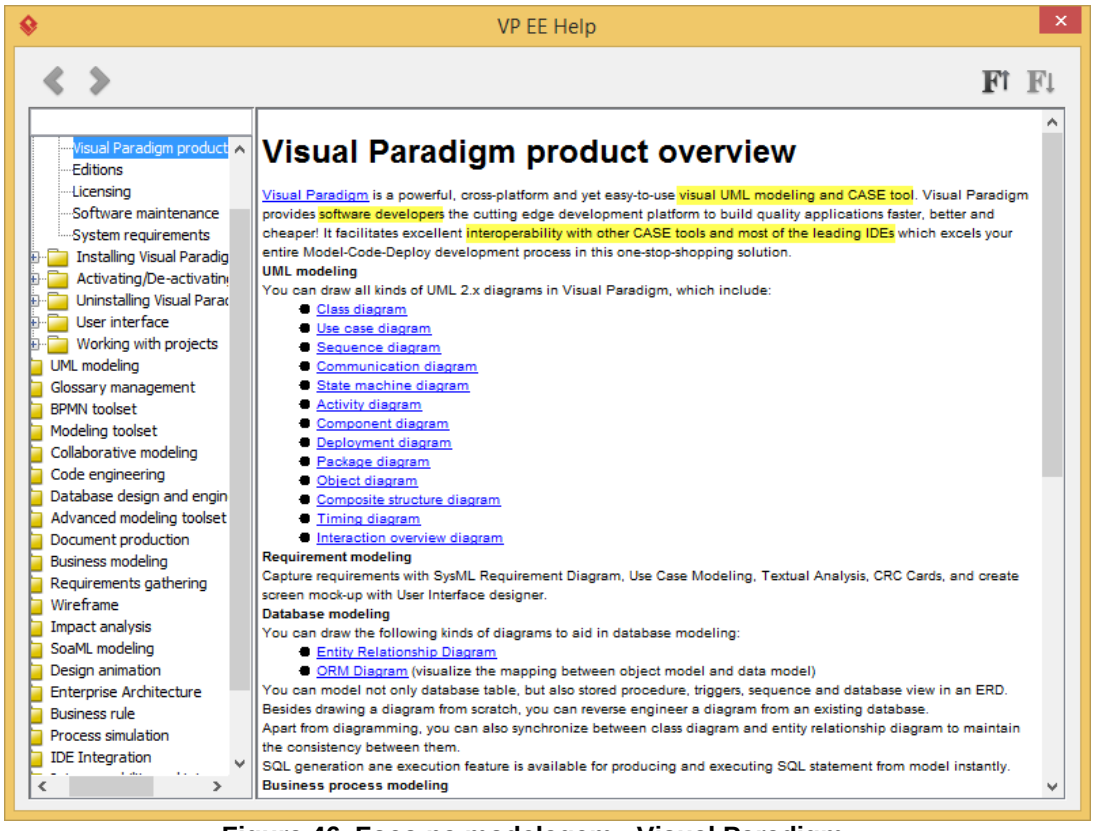


Figura 46. Foco na modelagem - Visual Paradigm

Estas evidências foram relacionadas à dimensão cognitiva **Consistência**, tendo o objetivo principal das ferramentas (modelagem) sempre destacado em diferentes pontos do manual, do website e do blog relacionados. Além das funcionalidades estarem, em sua maioria, condicionadas a existência de um modelo (Mnot) previamente criado.

4) Na subcategoria ***Modelo sempre em construção*** destacamos evidências relacionadas às funcionalidades que possibilitam a construção do modelo (Mnot) aos poucos, permitindo salvar versões ainda incompletas, segundo a UML. Por exemplo, podemos salvar o diagrama de atividades sem um nó inicial (Figura 47), o que não faz sentido, já que sem um nó inicial, não é definido onde o fluxo de atividades se inicia. O mesmo raciocínio vale para a ausência de um nó final ou mesmo de uma atividade, já que um diagrama de atividades sem atividades não faz sentido.

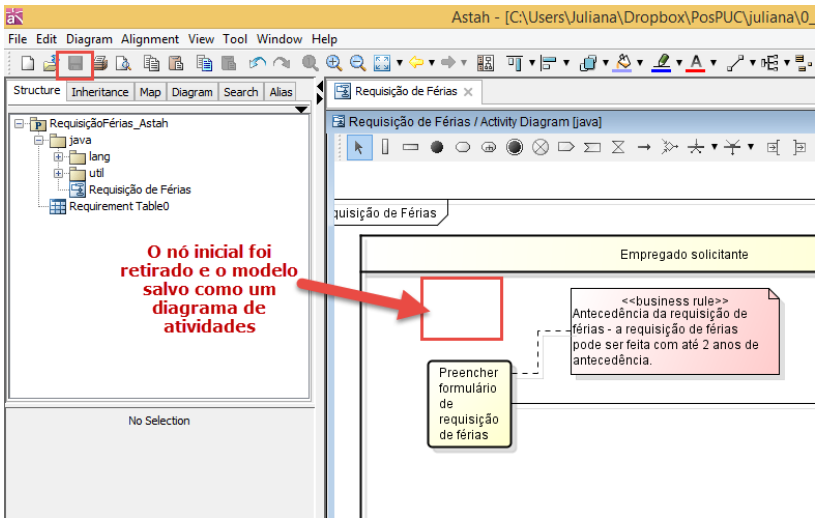


Figura 47. Diagrama de atividades sem nó inicial – Astah

Essa liberdade de ter sempre o modelo (Mnot) em construção possibilita que o produtor do modelo possa construir o modelo como preferir. Por exemplo, ele pode primeiro representar todas as atividades (Figura 48) e depois pensar como elas vão se relacionar no modelo.

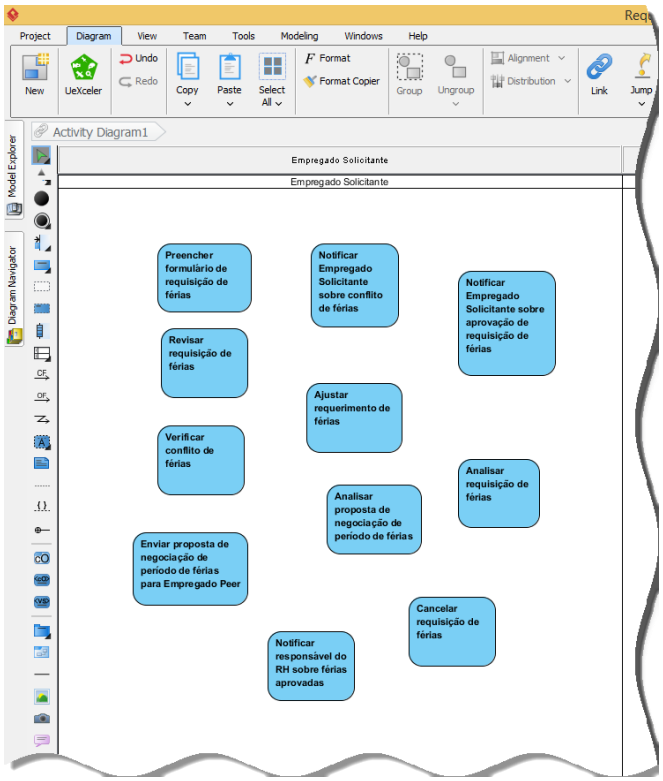


Figura 48. Todas as atividades criadas de uma vez - Visual Paradigm

Essas evidências foram relacionadas às dimensões cognitivas **Avaliação progressiva** e **Provisoriedade**. A possibilidade de construir modelos (Mnot) incompletos permite que o usuário vá construindo seu modelo aos poucos, e verificando o seu andamento conforme avança. Esta característica da notação da

ferramenta está relacionada à dimensão **Avaliação progressiva**. Não há necessidade de o modelo estar completo de acordo com a notação UML. O modelo pode estar em qualquer estágio que será salvo na ferramenta para evolução ou verificação futura. Essa característica de ter sempre os modelos em construção, também permite que o usuário tenha flexibilidade para brincar com ideias, mudar elementos de lugar, alterar a disposição dos elementos, tentar outras propostas de modelagem, flexibilidade essa que caracteriza a dimensão **Provisoriedade**.

Essas evidências foram relacionadas aos fatores ferramenta (T) e pessoas (P), onde a ferramenta (T) dá suporte a pessoas (P) para construir modelos (Mnot) aos poucos, sem a necessidade de ter um modelo completo para salvar o que foi modelado, podendo retomar a modelagem de onde parou em outro momento.

5) Na subcategoria ***Suporte à colaboração e trabalho em equipe*** destacamos evidências relacionadas às funcionalidades de colaboração entre membros da equipe de desenvolvimento (*team*) no processo de produção e consumo dos modelos de software através da ferramenta de modelagem (Figura 49).

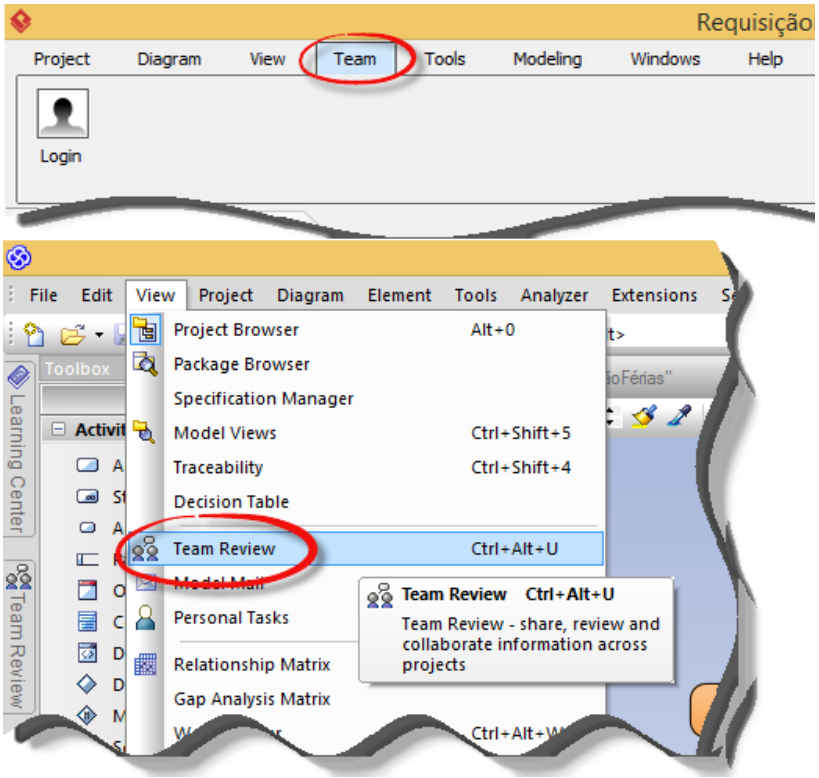


Figura 49. Funcionalidades de "Team" – VP e EA

Todas as ferramentas avaliadas apresentam recursos de colaboração, seja através de uma funcionalidade da própria ferramenta, ou com uma ferramenta

complementar ou *plug-in* associado. As evidências relacionadas ao suporte da ferramenta às diversas fases do processo de desenvolvimento (Figura 50) também comunica a colaboração entre pessoas, realizando as atividades relacionadas ao modelo em cada uma das fases (ex. Geração de código, Engenharia de Banco de dados, etc.).

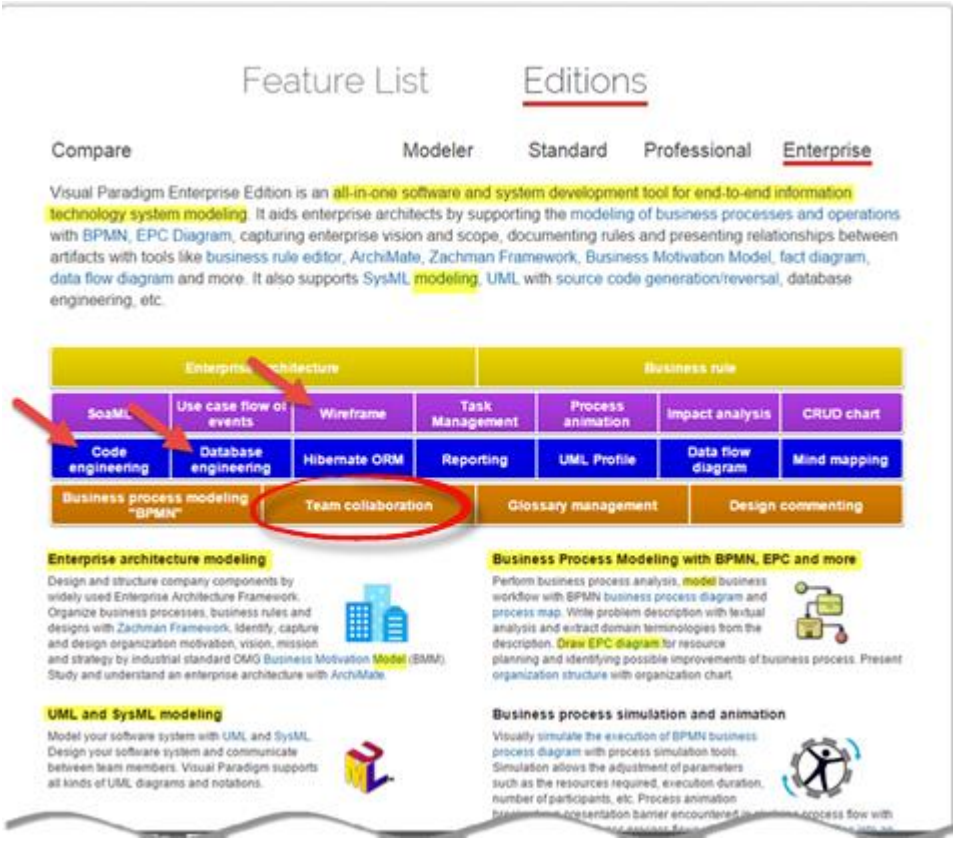


Figura 50. *Team collaboration* e suporte ao desenvolvimento - VP

As evidências desta subcategoria foram relacionadas à dimensão cognitiva **Expressão de papéis.** O *designer* contextualiza a atividade de modelagem dentro do processo de desenvolvimento de software, um contexto colaborativo onde pessoas trabalham em equipe (Figura 50). Ele coloca o modelo como um artefato deste processo, que está relacionado a outros artefatos (ex. código-fonte, banco de dados e etc.).

Essas evidências foram relacionadas aos fatores ferramenta (T) e pessoas (P), onde a ferramenta (T) dá suporte a pessoas (P) para construir modelos (Mnot) e outros artefatos gerados a partir dos modelos que fazem parte do processo de desenvolvimento de software. As funcionalidades de colaboração também têm a

relação T-P onde as pessoas (P) colaboram através da interação com a ferramenta (T).

Na categoria **Problemas de interação com modelos utilizando, principalmente, o mouse** definimos duas subcategorias análogas, sendo uma para a produção (***Produção de modelos utilizando, principalmente, o mouse***) e outra para o consumo do modelo (***Consumo de modelos utilizando, principalmente, o mouse***). Esta categoria tem relação com a subcategoria ***Consumidor pode fazer o mesmo que o produtor*** da categoria **Problemas no suporte ao processo de desenvolvimento de software**. A forma principal de interagir com a ferramenta para a produção ou consumo de modelos, é através do *mouse*. O teclado é usado para atribuição de nome e propriedade aos elementos e alguns atalhos. Porém, a ação de dispor os elementos e associá-los no modelo (Mnot) é realizada através do *mouse*, selecionando os elementos desejado na barra de elemento ou similar, e adicionando-os ao modelo (Mnot) em construção (Figura 51).

As evidências desta subcategoria foram associadas à dimensão cognitiva **Visibilidade**. Os elementos para a produção do modelo (Mnot) estão todos na régua ou galeria disponibilizada quando o usuário pede para abrir um modelo. Através do uso do *mouse*, o usuário pode selecionar o elemento e clicar no modelo para cria-lo ou arrastar o elemento até a área de modelagem. Todos os elementos disponíveis para aquele modelo estão visíveis, em alguns casos agrupados (Figura 52), para o usuário realizar a atividade pretendida. O teclado é usado para atribuir nome aos elementos e outras propriedades e também tem atalhos relacionados a algumas ações que são apresentados ao lado dos itens de *menu* relacionados.

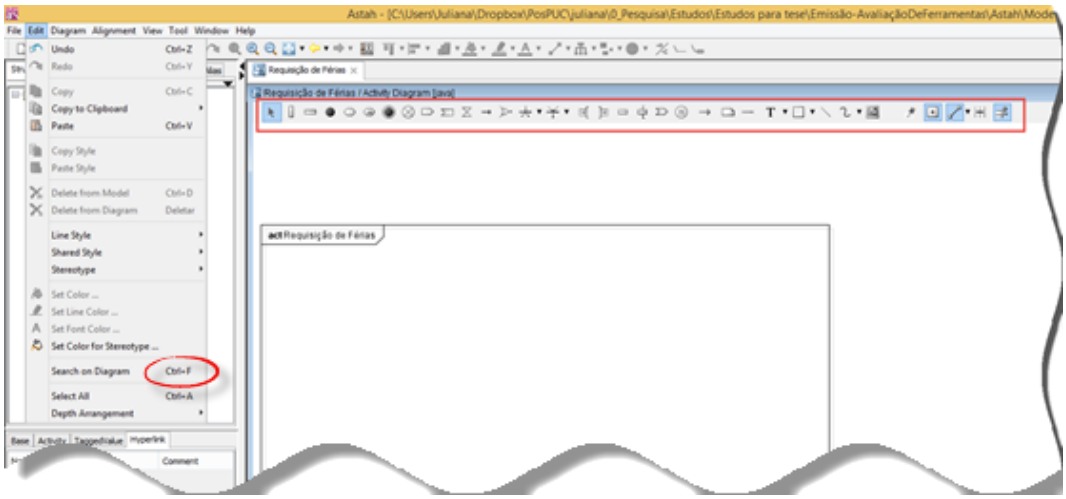


Figura 51. Interação principalmente usando o mouse - Astah

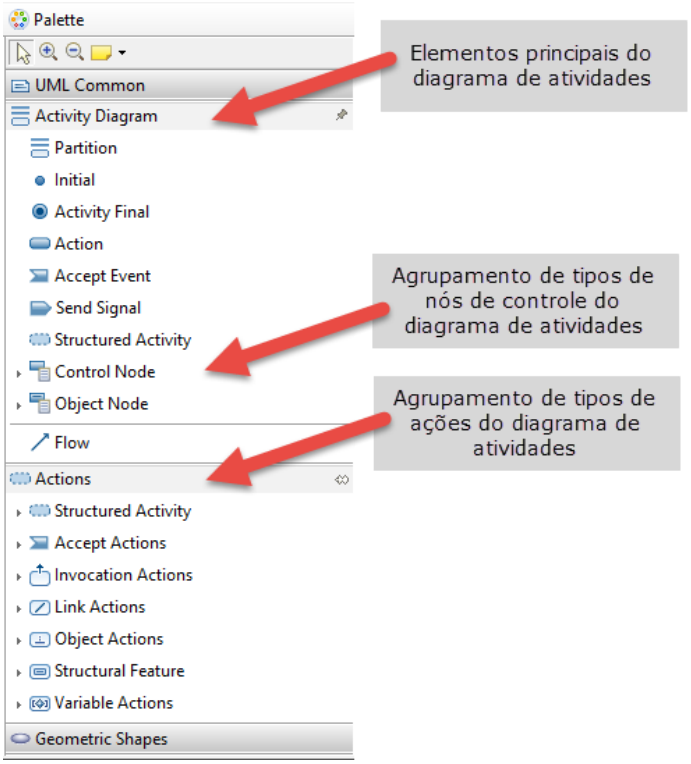


Figura 52. Agrupamento de tipos de elementos do diagrama

Essas evidências foram relacionadas aos fatores ferramenta (T) e pessoas (P), onde a ferramenta (T) oferece suporte a pessoas (P) para interagir com modelos durante as atividades de produção e consumo de tais modelos, utilizando o *mouse* e o teclado, para algumas ações.

5.2.3
Conclusão sobre Avaliação de Combinada

Nesta seção apresentamos uma conclusão sobre a comunicabilidade e usabilidade de ferramentas de modelagem, considerando as avaliações das 4 (quatro) ferramentas selecionadas.

Considerando os 2 cenários de avaliação, alguns achados foram comuns entre os 2 cenários de avaliação, particularmente relacionados aos signos metalinguísticos que apresentam uma mensagem mais focada no contexto onde a atividade de modelagem está inserida, o processo de desenvolvimento de software, do que na atividade de modelagem em si.

Os *template*s de metacomunicação relacionados aos signos metalinguísticos, estáticos e dinâmicos são bastante consistentes entre si (ex. todos apresentam evidências de que o *designer* entende que o usuário conhece bem a notação UML, usada no cenário de avaliação). A ausência de discrepância entre os *template*s evidencia que o *designer* tem uma mensagem bem determinada e clara, e reforça essa mensagem utilizando os três tipos de signos.

O contexto da mensagem de cada tipo de signo é o que difere mais em cada um dos *template*s. No caso do *template* composto com os signos metalinguísticos, a mensagem do *designer* é contextualizada considerando o processo de desenvolvimento como um todo, onde a produção de modelos é apenas uma das etapas deste processo e o modelo produzido pode ser consumido, de alguma forma (ex. evoluído, transformado, compartilhado). Além disso, no contexto do processo de desenvolvimento, o *designer* comunica que entende que o usuário pode ser um desenvolvedor que queira expandir a ferramenta de modelagem, então são apresentados recursos para que isso seja possível (uso e desenvolvimento de *plug-ins*).

No caso do *template* composto com os signos estáticos, a mensagem do *designer* ainda considera o contexto do processo de desenvolvimento (ex. indicando opções de exportação, de geração de código e outras possibilidades de ações a partir do modelo produzido). Porém, a mensagem fica mais restrita aos recursos que a ferramenta já oferece, recursos criados pelo *designer*. O *designer* pode indicar da possibilidade de expansão da ferramenta, (ex. item de *menu* "*Plug-in* list" no Astah), mas para evoluir sobre isso, somente através do uso de signos metalinguísticos fora da ferramenta em si (Explicações e tutorial no website).

No caso do *template* composto com os signos dinâmicos, a conversa entre *designer* e usuário fica focada na atividade de modelagem em si, não considerando o contexto maior do processo de desenvolvimento. Durante a produção ou consumo do modelo a mensagem do *designer* fica concentrada nas possíveis ações e reações relacionadas ao momento de interação do usuário com o modelo (ex. associação de elementos permitidos ou não, signos de redimensionamento de elementos, recursos de edição de representação, etc.). No momento da produção do modelo em si, o *designer* mantém sua mensagem focada naquela atividade que o usuário está realizando. Esta mesma mensagem se repete quando o usuário está consumindo o modelo. O *designer* oferece recursos extras (criação de um modelo de referência) para gerar um modelo não editável, que poderia eliminar as rupturas para relacionadas às funcionalidades de produção do modelo quando o mesmo é consumido. Porém, a utilização desse recurso depende da relação entre produtor e consumidor do modelo. O *designer* em si, indica o foco na produção do modelo.

O *template* de metacomunicação completo que representa a mensagem do *designer* para o usuário da ferramenta de modelagem de software, é construído a partir da combinação dos *template*s de cada signo. A análise foi realizada considerando as ferramentas selecionadas como uma classe de ferramentas. Portanto, consolidamos um só *template* de metacomunicação que representa a metacomunicação reconstruída dessa classe de ferramentas:

Eis a minha visão de quem você, ...”

Você conhece bem a notação UML, quem é responsável por sua especificação e suas várias versões. Você trabalha em equipe, é um desenvolvedor de software ou alguém interessado em desenvolver softwares e utiliza diversas notações para construir modelos, mas principalmente a notação UML.

“…o que aprendi que você deseja ou precisa fazer, de que formas preferenciais e por quê. ...”

Precisa, principalmente, de suporte para construção de modelos, mas também deseja ou precisa compartilhar os modelos que construir de alguma forma (gerando uma imagem do modelo, um XML, HTML, CSV, RTF, etc.) e

deseja ou precisa transformar os modelos que construir de alguma forma (gerando código, exportando para outras ferramentas, gerando banco de dados, etc.). Pode desejar ou precisar de representações extras (formas geométricas, textos livres, anotações/marcações à mão livre) durante a produção do modelo. Também deseja ou precisa construir modelos aos poucos e precisa ou deseja alterar a representação visual dos elementos do modelo. Também deseja ou precisa adaptar a ferramenta às suas necessidades, expandindo suas funcionalidades através do desenvolvimento de plug-ins ou uso de plug-ins já existentes.

"…Este é o sistema que consequentemente elaborei para você, e esta é a forma como você pode ou deve usá-lo..."

Portanto, aqui está uma ferramenta com o objetivo principal de apoiar a modelagem, usando a notação UML e outras notações, porém que oferece funcionalidade de compartilhamento e transformação dos modelos construídos em outros artefatos. Também ofereço a possibilidade de desenvolver plug-ins para estender as funcionalidades apresentadas na ferramenta. O mouse é o principal dispositivo para produção e consumo dos modelos, combinando com o uso do teclado para atribuir nome aos elementos e propriedades dos elementos criados e algumas ações nos elementos do modelo, como alinhamento de elementos, através de atalhos do teclado. O modelo pode ser salvo, mesmo que incompleto, pra ser construído aos poucos. Alguns elementos podem ter mais de um significado, e entendo que você seja capaz de fazer essa distinção e atribuir o significado correto durante a construção do modelo. Também ofereço a você a possibilidade de alterar a cor de fundo, cor da linha, cor da fonte, adicionar um ícone aos elementos apresentados, assim como criar elementos usando recursos de desenho, podemos associar os desenhos ao modelo.

"…para realizar um conjunto de objetivos que se enquadram nesta visão."

Para apoiar várias atividades do processo de desenvolvimento de software (modelagem, construção de banco de dados, codificação), começando pela modelagem do software e também para expandir as funcionalidades da ferramenta de modelagem, através do uso e desenvolvimento de plug-ins. Para que você tenha mais opções para criar a representação visual do modelo. Para

que o modelo seja complementado posteriormente quando necessário até que você considere o modelo finalizado. Para criar diferentes representações visuais dos elementos disponíveis no modelo em construção, possibilitando a distinção entre elementos com significados diferentes, atribuindo a cada um, uma aparência diferente.

A avaliação de usabilidade foi realizada investigando as porções da ferramenta relacionada às evidências de problemas de comunicabilidade identificados. As características cognitivas relacionadas à representação visual do modelo de software (Mnot) foram identificadas na avaliação de usabilidade das ferramentas de modelagem. Os recursos para visualizar a representação do modelo (**Visibilidade**), em partes ou por completo, considerando as relações entre as partes (**Expressão de papéis**) e os diferentes níveis de detalhamento e intenção dos modelos (**Abstração**) devem ser pensados para que tais relações fiquem explícitas para o usuário (**Dependências ocultas**) se apropriar das notações oferecidas pelo *designer* da ferramenta para produção e consumo de modelos. As diferentes representações disponíveis nas ferramentas de modelagem, assim como os recursos para criação de novas representações (**Notação secundária**), oferecem ao usuário a possibilidade de explorar ideias e possibilidades de modelagem (**Provisoriedade**), buscando uma representação (Mnot) que expresse o significado intencionado (Msem) pelo produtor do modelo. A produção de modelos de software, considerando aspectos da representação (Mnot) e da semântica (Msem), é um processo intelectual de construção de significado através de um artefato que tem os desafios da própria natureza de tal processo. Portanto, a ferramenta de modelagem usada neste processo deve apoiar o usuário nesta atividade evitando ações que demandem mais esforço cognitivo por parte do usuário (**Operações mentais difíceis**) em ações mais simples desse processo de modelagem como tratar de restrições da UML (**Propensão a erros**), por exemplo. Se a ferramenta mais atrapalhar do que ajudar na produção desse artefato intelectual, esta pode ser uma razão para desistir da ferramenta. Durante esse processo intelectual, o usuário deve ter recursos para verificar o andamento, o estado atual do artefato em produção (**Avaliação progressiva**) e tomar decisões a partir do artefato para seguir com sua construção. A característica cognitiva das ferramentas de posicionar modelos de software como parte do processo de desenvolvimento

(**Abstração**) oferece a oportunidade ao usuário em pensar no objetivo ou intenção do modelo (Msem) antes da produção ou consumo do modelo (Mnot).

5.2.4
Caracterização da tripla TNP

Considerando os três fatores da tripla TNP, podemos fazer as seguintes observações e relações entre cada um deles a partir dos achados destacados:

- O *designer*, representado pela ferramenta, sempre considera que o usuário é um profundo conhecedor da notação UML, suas limitações e versões. T disponibiliza os elementos da UML (N) relacionados ao modelo (Mnot) desejado, considerando que o produtor do modelo (P) conheça bem a especificação de N e suas restrições. Neste ponto temos os fatores T-N-P combinados. As decisões de T em relação à N considerando o P indicado estão relacionadas às dimensões cognitivas de **Visualização, Notação secundária** e **Propensão a erros.**
- O contexto onde a produção e consumo do modelo estão inseridos é amplamente comunicado por T. A modelagem é parte de um processo, onde o modelo será compartilhado com outras pessoas (P) e transformado em outros artefatos de desenvolvimento de software. Neste ponto temos os fatores T-P combinados, sendo que P tem duas representações: o P que produz o modelo e o P que vai, de alguma forma, consumir o modelo ao longo do processo de desenvolvimento. O posicionamento da modelagem no contexto de desenvolvimento de software apresenta características cognitivas das dimensões **Abstração**, onde existem diversos modelos (Mnot) com diferentes intenções e consumidores (Msem) ao longo do desenvolvimento; e **Expressão de papéis** onde a relação entre os diferentes modelos e, também, entre as pessoas (P) que produzem ou consomem tais modelos pode ser apoiada por T.
- A modelagem de software usando a notação UML (N) é o foco de T. No momento da construção de modelos, T provê os recursos à P para que faça uso de N para construir o modelo, podendo combinar a notação UML (N) com outras notações oferecidas. Neste ponto, temos os fatores T-N-P combinados. As diversas notações pré-definidas e recursos para estender tais notações indicam características de **Visualização** e **Notação Secundária**. T oferece recursos para combinar e estender N de tal forma que P possa produzir

modelos (Mnot) representando o significado necessário (Msem). Essa flexibilidade oferecida por T está relacionada também a característica de **Proximidade de mapeamento,** já que no mundo real a produção de modelos pode misturar notações existentes ou mesmo criar notações para construir modelos.

- Quando o *designer* (T) oferece recursos para expandir a ferramenta através do desenvolvimento de *plug-in*, ele está se comunicando com um colega *designer* que é seu usuário (P). O usuário é alguém de tem a capacidade ou mesmo vontade de desenvolver software, e para isso, o *designer* oferece o apoio que acha necessário. Neste ponto, temos T-P combinados e relacionados com a característica cognitiva de **Notação Secundária.** T provê recursos para que P estenda a sua notação como um todo (da ferramenta) e, em alguns casos, da notação de modelagem também.
- Em algumas situações, o *designer* (T) lembra seu usuário (P) que é grande conhecedor da UML (N) sobre as restrições da notação. Porém, essa ajuda não está sempre disponível, deixando toda a carga de garantir o alinhamento com a especificação da notação para o usuário. Mesmo conhecendo bem a notação, o usuário pode cometer algum deslize e não perceber no mesmo momento, ou desejar fazer uma verificação em um modelo que considere finalizado. T poderia oferecer recursos para verificar mais amplamente N e no momento que P necessitasse. Neste ponto, temos os fatores T-N-P combinados. A característica cognitiva aqui relacionada é **Propensão a Erro,** já que o suporte em relação à UML (N) oferecido por T não é completo e não está sob o controle do usuário (P).
    - O *designer* (T) oferece a possibilidade de P expandir visualmente a notação do modelo (N), alterando cores, incluindo ícones, imagens, texto. Essa possibilidade de expandir a notação pode ser muito benéfica para a comunicação usando os modelos. Porém, o significado desses novos elementos de notação do modelo não fica claro, o que pode provocar problemas para quem for consumir o modelo. Novamente, fica nas mãos do usuário atribuir e transmitir esse significado às notações criadas de alguma forma. Neste ponto, temos os fatores T-N-P combinados. A característica cognitiva de **Proximidade de mapeamento**, já que T dá o suporte para que N seja flexível e variante, enquanto usada por P. Essa flexibilidade é semelhante

àquela do processo de modelagem feito sem apoio de T. Também temos a característica cognitiva de **Notação secundária**, já que novas representações podem ser criadas e, associada a esta, outra característica cognitiva de **Dependências ocultas** em relação aos novos elementos de representação da notação. Muitas vezes o significado atribuído por P aos novos elementos de N não fica relacionado à T, sendo usado um protocolo social para atribuir significado aos novos elementos. A dependência entre representação (Mnot) e significado (Msem) pode não ficar clara para os P consumidores dos modelos criados usando os novos elementos.

- O *designer* (T) possibilita que o usuário (P) brinque com N durante a construção do modelo. Um modelo (Mnot) não precisa estar completo para ser representado e salvo, dando espaço para a criatividade do usuário P que cria o modelo, e possivelmente outros Ps que possam ter interesse no modelo, em pensar, trocar ideias, testar diferentes representações de um significado usando o modelo como objeto de representação. Quem define quando um modelo está pronto, é quem o constrói. T dá liberdade a P para usar N até que considere que o modelo está completo ou finalizado. Neste ponto, temos os fatores T-N-P combinados. As características cognitivas de **Avaliação progressiva** e **Provisoriedade** estão relacionadas com o suporte que T oferece para P explorar e experimentar com N até que decida que o modelo está finalizado.

5.3
Triangulação dos Resultados

Nesta fase validamos os estudos realizados através da comparação e discussão dos resultados das fases de investigação do contexto de desenvolvimento de software e interação com ferramentas de modelagem de software. Além do contraste entre estes resultados das fases de investigação do contexto de desenvolvimento de software e interação com ferramentas de modelagem de software, também fazemos o contraste com dados de pesquisas anteriores. A TNP também é usada como recurso de articulação da triangulação. A partir da análise das categorias e evidências de cada uma das fases, identificamos as seguintes situações:

Relação direta entre categorias relacionadas ao contexto de desenvolvimento e relacionadas a interação de ferramentas de modelagem. Esta situação nos

possibilitou fazer a triangulação de evidências identificados na investigação do contexto de desenvolvimento com evidências da investigação da interação de ferramentas de modelagem, sendo possível associar problemas e questões identificadas junto aos profissionais sobre ferramentas de modelagem de software à problemas e questões de comunicabilidade e usabilidade de tais tipos ferramentas.

Relação mediada entre categorias relacionadas ao contexto de desenvolvimento e relacionadas a interação de ferramentas de modelagem. Esta situação foi verificada para categorias de problemas relacionadas ao contexto de desenvolvimento que não tiveram uma relação direta com categorias relacionadas a interação de ferramentas de modelagem. Entretanto, tais categorias possuíam relação com outras categorias relacionadas ao contexto de desenvolvimento, estas sim relacionadas à categoria relacionada ao contexto de desenvolvimento (Figura 53).

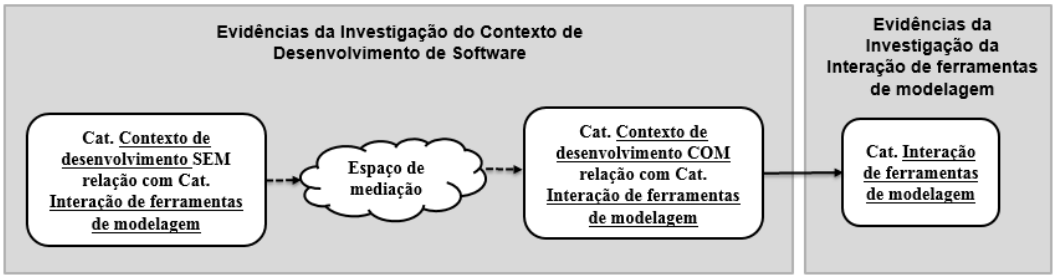


Figura 53. Esquema da relação mediada

Como essa mediação não foi formalmente evidenciada é necessária uma investigação específica esse espaço de mediação. Entretanto, as características das evidências de cada categoria forneceram informações suficientes para que a relação indireta fosse estabelecida. As categorias relacionadas ao contexto de desenvolvimento sem relação direta com categorias relacionadas a interação de ferramentas de modelagem estão apresentadas na Tabela 7.

Essa situação de mediação foi identificada devido ao diferente nível de abstração das duas fases de investigação da pesquisa, onde na primeira fase consideramos o contexto com e sem apoio de ferramentas de modelagem de software, e na segunda foram realizadas avaliações de quatro dessas ferramentas. Mesmo sem dados aprofundados sobre os espaços de mediação, a identificação dos mesmos indica caminhos de pesquisas a serem realizadas sobre modelos de software.

Tabela 7. Categorias relacionadas ao contexto de desenvolvimento sem relação direta

Categoria Contexto de Desenvolvimento de Software	Subcategoria
Contexto do Processo de produção e consumo do modelo	Abandono de modelos
	Confiança nos modelos
	Modelos devem ser úteis
	Padrão de produção de modelos
	Produtores de modelos são consumidores exigentes de modelos
Modelagem influenciada pelo contexto de desenvolvimento	Nível de abstração de modelos
	Ponto ótimo do modelo

Um exemplo de relação mediada foi identificado entre a subcategoria relacionada ao contexto de desenvolvimento ***Produtores de modelos são consumidores exigentes de modelos*** e a categoria **Ferramenta influenciando/impactando modelagem** (Figura 54) que tem relação direta com a subcategoria de investigação de interação de ferramentas de modelagem ***Diversas opções de menu para exportar e importar de/para diversos formatos***.

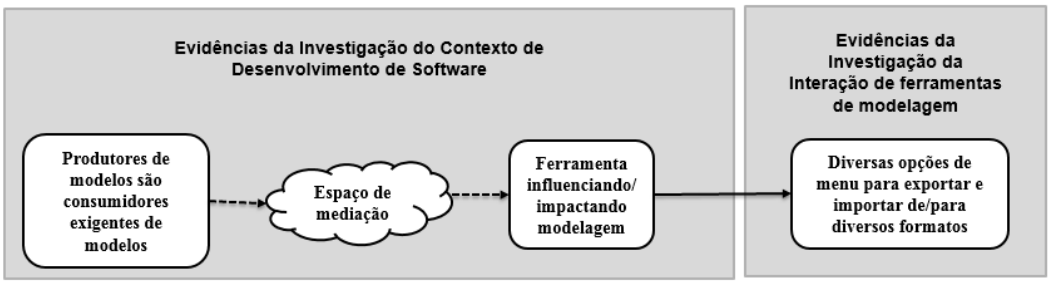


Figura 54. Exemplo de relação mediada

Considerando o contexto do processo de produção e consumo de modelo de software, podemos argumentar que o consumo do modelo, seja quem for esse consumidor, pode ser influenciado pela ferramenta de modelagem usada na produção do modelo. Se a ferramenta oferecer bons recursos para construção e compartilhamento do modelo, só vai depender do produtor a construção de um bom modelo. Entretanto, se a ferramenta oferecer recursos ruins ao produtor do modelo, este vai ter que construir o modelo dentro daquelas limitações. Por exemplo, a ferramenta pode somente exportar o modelo para um formato onde o

modelo fique distribuído em diversas páginas, ou que no processo de exportação/importação do modelo ele perca algumas características atribuídas pelo produtor. Quando o modelo chegar ao consumidor, que neste cenário é particularmente exigente por ter experiência na construção do dado modelo, os problemas causados pelos recursos de exportação/importação podem influenciar o consumo do modelo.

De qualquer forma, este espaço de mediação é um espaço que necessita de investigação específica. No contexto desta pesquisa só tivemos informações suficientes para apresentar a relação mediada, ficando esse espaço de mediação em aberto para futuras pesquisas. A listagem completa da relação entre categorias duas fases de investigação está apresentada no Apêndice F.

Um fato interessante sobre a relação por mediação está relacionado à categoria **Modelagem influenciado pelo contexto de desenvolvimento**. Esta é a categoria que mais aparece como mediadora para categorias de contexto de desenvolvimento sem relação direta com categorias de interação com ferramenta de modelagem. Como a descrição da própria categoria indica, ela reúne questões que tem um foco maior no fator P da tripla, considerando o contexto social (de desenvolvimento de software) onde a modelagem ocorre. A investigação do contexto de desenvolvimento, considerando pessoas, práticas e outras perspectivas tem sido a indicação de outras pesquisas recentes relacionadas a modelos de software (Petre, 2013) (Whittle et al. 2013) (Whittle et al., 2014).

Um achado interessante é a presença de outros P's quando a construção de modelos é considerada como parte do processo de desenvolvimento, já que o modelo (Mnot) pode ser input para outras atividades (ex. geração de código, banco de dados, etc). O *designer* deixa claro através da interface da ferramenta de modelagem que aquele modelo sendo construído é um artefato que, possivelmente, será evoluído e transformado pelo próprio construtor, ou, como normalmente é o caso, por outras pessoas da equipe de desenvolvimento.

Tendo a Engenharia Semiótica (de Souza, 2005) como teoria base e o Método de Inspeção Semiótica como parte do método combinado para a avaliação das ferramentas, os pontos em que temos T-P combinados, mesmo dentro da combinação T-N-P, são particularmente interessantes. Na verdade o que temos é a relação entre P's: *Designer* da ferramenta (T), representado pela interface de T durante a interação, e o usuário (P) da ferramenta. Tendo em mente essa relação

entre P's, quando são associados *plug-ins* a uma ferramenta, um único diálogo (*designer*-usuário) se transformar numa série de diálogos entre um mesmo usuário e diferentes *designers*, um para cada *plug-in* associado, por exemplo, reunidos em uma mesma interface. O efeito no limite pode ser o de uma colcha de retalhos de comunicação, cada *plug-in* com padrões próprios e distintos dos padrões dos demais e dos padrões da própria ferramenta.

A adição de um *plug-in* pode causar um efeito tão simples quando o surgimento de um novo item de *menu* ou tão complexo quanto a integração de toda uma nova funcionalidade, com novas metas associadas, novas interações, novos controles de interface, e assim por diante. Na Figura 55 temos um exemplo de como a conversa com mais de um *designer* pode ficar confusa para o usuário. Dois *plug-ins* diferentes foram instalados no Astah, ambos relacionados ao formato XMI (padrão da OMG para troca de informações baseado em XML), porém o *designer* de cada um dos *plug-ins* estruturou e comunicou a sua mensagem de forma diferente para usuário, provocando um ruído na interação final do usuário com a ferramenta estendida pelos *plug-ins*: o menu tem uma entrada XMI e, enquanto a importação é comunicada em um *submenu* de XMI, a exportação é comunicada como um novo item de mesma ordem que XMI, que, pela lógica, deveria estar no mesmo nível que a importação. Uma situação semelhante foi apresentada por de Souza em Leitão (2009) no contexto de desenvolvimento *open-source*, onde o desenvolvimento é realizado por diversos desenvolvedores voluntários e a articulação entre eles para garantir que a mensagem de metacomunicação fique consistente é um grande desafio. Quando a ferramenta oferece a possibilidade de desenvolver *plug-ins*, todo usuário é, potencialmente, um *co-designer* dessa ferramenta, o que deixa a mensagem de metacomunicação original do *designer* da ferramenta exposta a interferências imprevisíveis. Este seria um ponto de atenção importante para a comunidade de *Model-Driven Software Engineering*, que está considerando o *open-source* como a única solução para modelagem na indústria. (Cabot, 2015a)

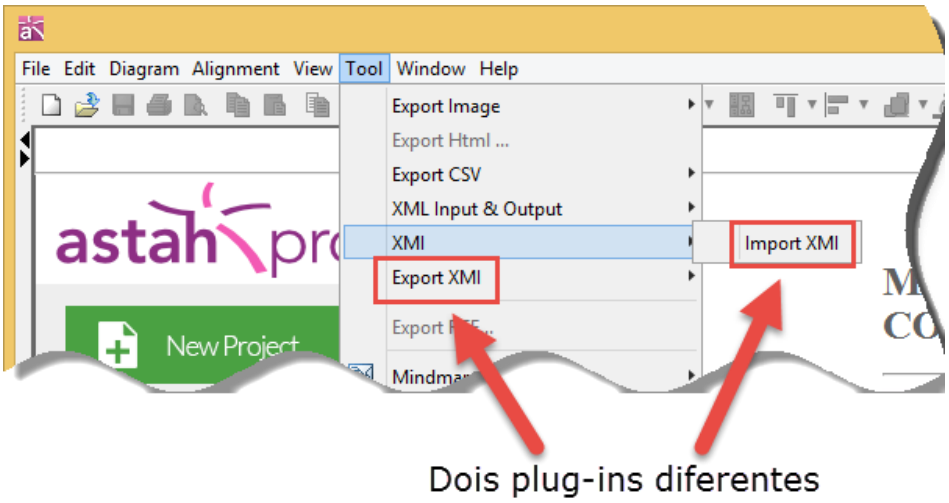


Figura 55. *Plug-ins* - Vários *designer*s conversando com o usuário

Durante os estudos, o papel do modelo como objetos de coordenação entre pessoas apresentou muitas evidências. Na subcategoria do contexto de desenvolvimento ***Objetos limítrofes do domínio como integradores***, relacionada à categoria de interação com ferramenta de modelagem ***Suporte à colaboração e trabalho em equipe***, os modelos foram caracterizados como peças de comunicação entre pessoas. Os conceitos importantes do domínio do problema relacionado ao software devem estar representados e diferentes etapas do desenvolvimento. Os objetos limítrofes (Star, 1990) (Lee, 2005) apresentam essa característica de estarem sempre representados de alguma forma, para servirem de referência para pessoas de perfis diferentes. Os modelos, conforme vão sendo evoluídos e transformados, devem dar conta desses conceitos importantes, os objetos limítrofes do domínio, para que conservem sua capacidade de comunicar com pessoas.

Relacionado à função de coordenação do modelo, também podemos associar as evidências que apontam para a necessidade de um protocolo social (Ellis, Gibbs, 1991) para complementar o protocolo tecnológico antes e ao longo da modelagem de software. Como a especificação da notação UML, mais utilizada para a modelagem de software, é bastante ampla, para cada contexto (empresa, projeto, equipe) deve haver uma definição prévia do que cabe para aquele contexto em relação a modelos (Mnot) necessários, nível de detalhamento e outras decisões que não são da alçada tecnológica, mas que impactam o uso da tecnologia (ferramentas de modelagem) e a construção de tecnologia (software desenvolvido). O contexto social tem grande força e impacto no processo de

desenvolvimento de software e deve ser reconhecido e investigado. (Whittle et al., 2014)

O modelo como representação do escopo de desenvolvimento, também foi evidenciado. A representação do domínio do software através de modelo é uma forma possível de definir fronteiras e coordenar quais pessoas serão responsáveis por cada parte do modelo (Mnot). A coordenação do trabalho a ser realizado (*i.e.* o desenvolvimento de software) é feita com o uso do modelo, como objeto de contexto, considerando conceitos do domínio de mais alto nível, ou até mesmo como objetos de tarefa, quando chega à fase de codificação (ex. quando dois desenvolvedores precisam manipular a mesma classe de dados). (Prates, 1998)

Nos achados relacionados à categoria do contexto de desenvolvimento "Modelos práticos *versus* Modelos para documentar", as evidências do contexto de desenvolvimento foram fortes em relação aos modelos poderem ser usados na prática, e não como documentação que acaba desatualizada. Porém, isso vai depender da função da documentação. No caso da geração de sistema de ajuda a partir de modelos, (Moriyon et al., 1994) (Pangoli e Paternò, 1995) modelos inconsistentes e desatualizados podem levar a sérios problemas de interação para os usuários finais do software final. Para esses casos, os modelos devem refletir exatamente o que foi desenvolvido em determinada versão do software, caso contrário, quem acaba sofrendo as consequências é o usuário final do software. Na Figura 56 temos um exemplo retirado de uma das ferramentas de modelagem avaliadas, onde o conteúdo do sistema de ajuda, relacionado a um caminho de interação com a ferramenta, não coincide com a interface da versão relacionada ao sistema de ajuda.

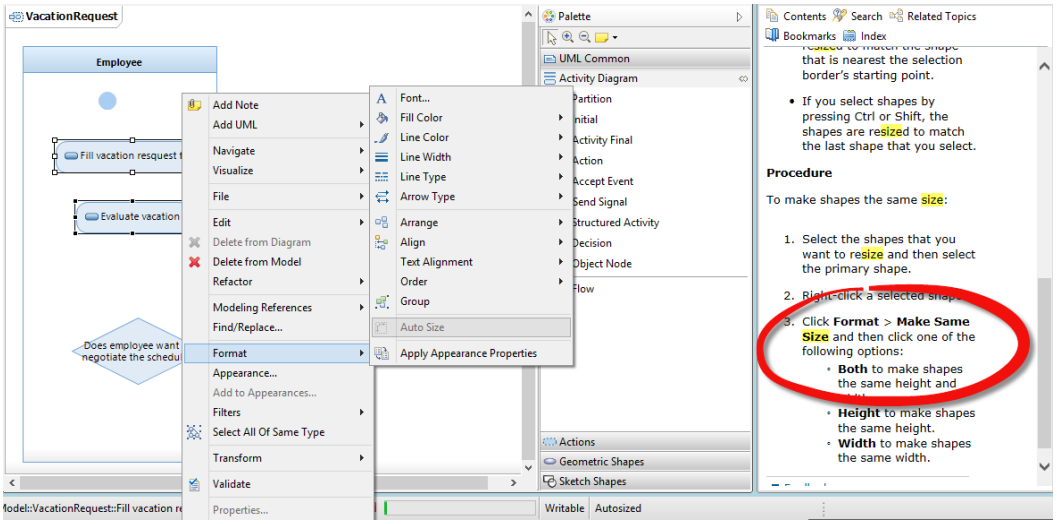


Figura 56. Opção "Make same size" não existe na interface

Outros achados relevantes estão relacionados à subcategoria do contexto de desenvolvimento ***Validação UML por ferramenta de modelagem*** e a subcategoria de interação com ferramenta ***Algumas restrições da UML são implementadas***, quando as ferramentas não oferecem uma validação UML pobre ao usuário. (Cabot, 2010) Mesmo que o usuário seja um especialista em UML, o recurso de verificação é útil para evitar lapsos e deslizes que são totalmente possíveis de acontecer. Realizar essa verificação ou não, deve ser escolha do usuário. Porém, nossos achados indicam que, quando existe a verificação, não fica claro para o usuário o que está sendo verificado. Portanto, caso o usuário pretenda usar este recurso para fazer uma verificação final no modelo, as ferramentas não oferecem uma validação completa da UML, deixando passar coisas simples como um elemento solto no modelo (ex. IBM RSA e Enterprise Architect). Também identificamos o caso onde existe a função de verificação, porém não é o usuário que define quando essa verificação será executada. O usuário ativa a opção e tem que 'esperar a ferramenta' fazer a verificação, apresentando os problemas no modelo (ex. Visual Paradigm). Esses são problemas de usabilidade das ferramentas que impactam sua comunicabilidade já que o usuário pode estar confiando numa validação parcial ou então ficar à mercê de uma verificação que ele não pode controlar quando será executada.

Como a execução do cenário de consumo do modelo, obtivemos evidências de que a ferramenta de modelagem tem foco na atividade de produção do modelo. As ferramentas apresentam diversas formas de compartilhar os modelos ali produzidos, porém o consumo através da própria ferramenta deve ser feito com

cautela, já que as funções usadas para a produção do modelo se encontram ativas. Alguns recursos, como a criação de um modelo de referência, foram identificados, porém a utilização destes tem uma mensagem implícita do produtor do modelo para o consumidor. O produtor do modelo, ao compartilhar um modelo *read-only*, comunica ao consumidor que ele entende que o consumidor é alguém que não tem capacidade de interagir com um modelo editável. Ou seja, a forma como a ferramenta é usada pode dar indicações sobre a relação entre as pessoas envolvidas no processo de modelagem de software, assim como sobre seu conhecimento da notação.

Os achados relacionados aos recursos de extensão da notação do modelo, inclusive sobre as extensões previstas na UML (estereótipos, tags e *constraints*), reunidas na categoria do contexto de desenvolvimento **Modelagem influenciada pelo contexto de desenvolvimento**, aponta para o que Eco (2000) chama de **hipercodificação e hipocodificação**. A **hipercodificação** acontece quando uma regra é adicionada para estabelecer uma aplicação particular de uma regra geral. Nos nossos achados, temos o caso do uso de *tags*, reportado pelo Participante 8, onde somente classes com a '*tag* gera código' eram consideradas para serem transformadas em código fonte durante o processo de desenvolvimento. O produtor do modelo era o responsável por atribuir esse rótulo ao elemento do modelo. A **hipocodificação** é realizada quando, na ausência de regras estabelecidas, porções de uma notação são tomadas como unidades pertinentes a um código em formação, capazes de veicular porções efetivas de conteúdo. Nos achados temos o uso do estereótipo da UML para representar a regras de negócio, que é uma porção de texto que descreve um comportamento que o software possivelmente deve implementar. A própria UML, com suporte das ferramentas, apresenta recursos para estender a notação de modelagem.

Considerando o processo de produção e consumo de modelos, a questão da subcategoria ***Conhecer UML versus compreender UML*** foi identificada na fase de investigação do contexto de desenvolvimento de software e relacionada à categoria de interação com a ferramenta **Problemas com recursos para construção de significados baseado somente em UML**. A ferramenta apresenta os elementos da notação UML para que o produtor construa o modelo, porém a mensagem que o modelo irá apresentar está relacionada à forma como o produtor do modelo vai estruturar o seu discurso em UML.

A produção do discurso em UML, o modelo, pode ser associado aos achados relacionados à subcategoria de contexto de desenvolvimento ***Conteúdo mais importante que forma***, onde os participantes indicavam que problemas na forma do modelo, relacionados à notação UML (Mnot) não são considerados tão graves quando de conteúdo (Msem). Esses achados apresentam questões relacionadas à **aceitabilidade** e **gramaticalidade** de uma linguagem. Estas noções são tomadas do trabalho de Chomsky (1971) ao teorizar sobre a linguagem natural. Ele define que a **gramaticalidade** é característica que um falante ideal de uma língua humana possui de falar bem e adequadamente, produzir sentenças que sempre podem ser descritas por uma gramática da língua. Já a **aceitabilidade** é o grau de aprovação que um falante real tem em relação às frases que ouve, que pode levar à aceitação de produções agramaticais. Quando temos evidências dos participantes dizendo que eles "*digerem qualquer coisa que vier*", ou relevam erros óbvios de UML, esse grau de aceitabilidade se mostra como um ponto crítico se contrastado com a gramaticalidade do modelo (Msem). Algumas incorreções podem ser corrigidas se o consumidor e produtor compartilharem o mesmo contexto (pela força de coerção do contexto). Por exemplo, se a construção do modelo está sendo feita de forma colaborativa, todos têm acesso ao contexto de produção e podem passar por cima de problemas de expressão. Entretanto, se o modelo está sendo usado com o transmissor da mensagem do produtor para o consumidor, sem ter o contexto de produção compartilhado, é o signo que vai ter que construir por si só o contexto, é o modelo que vai ter que expressar e reconstituir o contexto para o consumidor.

Ainda devem ser considerados casos onde o contexto não possibilita a correção do erro ou o receptor não tem nem a capacidade de identificar que existe uma incorreção. Quando a definição de certo e errado do consumidor é diferente daquela do produtor, a incorreção pode passar despercebida. No caso de modelos UML, existe um ponto de convergência que é a especificação da notação feita pela OMG. Porém, o que está relacionado ao conteúdo do modelo (Msem) não possui esse ponto comum. A convergência do significado vai depender daquilo que é 'correto' para o consumidor comparado ao o que é 'correto' para o produtor em relação ao modelo representado (Mnot).

Para qualquer linguagem, a capacidade de compreender um discurso antecede a capacidade de emitir um discurso nessa mesma linguagem. (Ingram,

1989) Conforme uma pessoa produz o discurso, ela também está realizando o processo de compreensão daquele discurso. Os processos de compreensão e produção se desenvolvem de modo paralelo, com a produção sempre acompanhando a compreensão. Para que alguém seja capaz de produzir um discurso correto precisa ter sido consumidor de discursos corretos naquela linguagem, ter experimentado com a linguagem, produzido discursos para compreender. A compreensão do discurso antecede e é a base da produção do discurso. Um aprendiz de uma determinada linguagem deve primeiro escutar o som do discurso antes de saber quais sons fazer e um aprendiz deve ouvir o som do discurso coordenado com a experiência junto a objetos, situações, eventos do ambiente ou da mente antes que a pessoa possa atribuir significado ao som do discurso. (Ingram, 1974) Ou seja, o contexto do discurso é de grande importância para que o mesmo seja propriamente recebido.

No contexto de modelagem de software, um produtor de modelos deve compreender a UML, e não simplesmente conhecer, de forma que tenha capacidade de estruturar o discurso através do modelo. Não é somente questão de relacionar elementos, mas de construir um artefato que vai transmitir ao consumidor daquele modelo o discurso que o produtor intencionou. (Petre, 1995) As ferramentas de modelagem de software, natureza de ferramentas avaliadas nesta pesquisa, não dão conta desta necessidade.

O desenvolvimento de software é um processo que envolve muito conhecimento técnico. Porém, o objetivo é construir uma solução tecnológica para solucionar problemas de um determinado domínio do mundo real. Os modelos têm um papel epistêmico, particularmente no início do desenvolvimento, onde o problema está sendo conhecido e discutido. Através da interação com o modelo, seja através de ferramenta ou mesmo sem (quadro branco, papel e etc.), as pessoas envolvidas no desenvolvimento da solução de software realizam o que Magnani (2005) chama de abdução manipulativa. A abdução manipulativa é um processo de inferência disparado pela manipulação de um objeto externo ao espaço intrínseco de um problema que a inferência está tentando resolver, no caso, o modelo. Ela acontece quando se pensa enquanto se faz algo com este objeto externo, mas não somente, em no sentido pragmático de fazer. Também é forte no sentido cognitivo (epistêmico), para conhecer melhor os espaços de problema, por exemplo. (Magnani, 2005) O modelo tem esse papel de apoiar e influenciar a

abdução, apresentando recursos e visões que apoiam o processo de entender o problema, e os possíveis caminhos de solução.

Nossos *insights* sobre algum assunto, que muitas vezes estão corretos a respeito de certos fatos, são resultados de um tipo de sexto sentido. Porém, tais *insights* são possíveis a partir de experiências anteriores, da combinação de conhecimentos prévios. (Santaella, 2005) Os modelos de software apoiam a abdução dos envolvidos, seja no processo de produção ou consumo. O modelo de software define um espaço, um contexto para que esses *insights* ocorram, sendo estes *insights* a combinação simultânea de inferência, a partir do que já se sabe, com algo novo e original. (Santaella, 2005)

Neste processo de pensar em possíveis soluções, primeiro é necessário que se conheça bem o problema, e o domínio relacionado, para que seja possível emitir qualquer discurso sobre este problema para o qual o software está sendo proposto como solução, ou parte dela. A epistemologia da prática descreve as formas de conhecimento e modos de saber que informam ou influenciam nossas ações. Para produzir conhecimento é necessário agir sobre o que já se sabe e aprofundar esse conhecimento. Toda ação neste caminho se explica como uma etapa para aproximação do objetivo final: um novo conhecimento. (Schön, 1983) Porém, nesse processo de construção da solução de problemas, podem ser realizadas ações que não têm foco no objetivo final (ações pragmáticas), mas sim para conhecer mais as condições de solução de problema (ações epistêmicas). (Kirsh e Maglio, 1994) É preciso interagir com o objeto para conhecê-lo melhor. Desta forma, quando for construir o objeto, as coisas que foram epistêmicas para o produtor para a construção do modelo, podem ser descontadas para o consumidor.

A modelagem de software tem como objetivo abstrair o que não é parte do domínio do problema, e aqueles que produzem os modelos fazem esse papel de definir o que precisa ser comunicado, e como, ao consumidor do modelo. Por isso o papel do produtor do modelo e o contexto de produção e consumo de modelos de software devem ser investigados, considerando todos os fatores relacionados - ferramentas (T), notação (N) e pessoas (P) - para que seja possível identificar como cada um desses fatores impacta nesse processo de construção das abstrações, de construção de conhecimento sobre o domínio do problema, para que o software, objetivo final, faça parte da solução do problema que motivou seu

desenvolvimento. Se cada artefato fizer o seu papel de representar e comunicar ao longo do processo de desenvolvimento do software, cada um dos envolvidos vai poder executar suas funções com menos "ruídos de comunicação": *"... eu não preciso me preocupar como eu vou abstrair aquele negócio, tá abstraído ali pra mim, só tenho que implementar aquilo... que seja bem feito, né? (risos)" (Participante 6).*

Como apresentado e discutido acima, além dos dados coletados nas duas fases de estudos (investigação do contexto de desenvolvimento de software e investigação da interação) e de referências formais de outras pesquisas (*journals*, artigos científicos e livros), utilizamos postagens do *blog MOdeling LAnguages* como fonte de dados para discutir e triangular os achados desta pesquisa.

O *blog MOdeling LAnguages* é mantido por Jordi Cabot, professor e pesquisador da área de MDSE e um dos autores do livro "*Model-driven Software Engineering in practice*" (Brambilla et al., 2012). Jordi Cabot também tem atuação no mercado, o que possibilita que suas postagens tenham foco na indústria, na academia ou buscando a ligação entre os dois domínios. Entendemos que os dados do *blog* podem indicar algum viés ou orientação para assuntos de interesse das pesquisas do autor, porém também acreditamos que são referências informadas sobre muitas questões e situações relevantes para esta pesquisa. Portanto, utilizamos postagens do *blog* como dados para triangulação com dados coletados durante nossa pesquisa, fomentando a discussão de nossos achados.

O pesquisador possui a conta *@softmodeling* na rede social *Twitter*, onde constantemente indica postagens antigas de seu *blog MOdeling LAnguages* com a *hashtag #oldiebutgoldie*. Essa classificação através da hashtag *#oldiebutgoldie* (numa tradução livre estaria classificando alguma questão antiga, porém ainda pertinente), nos levou a pensar se tais questões abordadas no passado e novamente referenciadas na rede social, ainda são observadas na prática ou se ainda são relevantes no contexto de modelagem de software. Um exemplo de postagem *#oldiebutgoldie* foi intitulada "*Poor validation of UML Models in Eclipse UML2 tools*" (Cabot, 2010). Esta postagem foi feita em Abril de 2010 e referenciada na conta do pesquisador no *Twitter* em dia 5 de Fevereiro de 2015. Nossos achados apontam algumas questões relacionadas à validação de modelos UML, ou seja, a nova referência ao assunto de validação pobre de modelos UML abordado em

2010, se mostrou como mais uma evidência de que esta questão ainda está em aberto, assim como outras abordadas no *blog*.

Entendemos que o uso dessa referência em particular deve ser destacado, já que o viés do pesquisador responsável pelo *blog* pode influenciar no viés da pesquisadora. Apesar de não ser uma referência formal, o *blog* se mostrou como uma interessante fonte de evidências que reforçaram ou complementaram nossos achados.

5.4
Sobre este capítulo

Neste capítulo apresentamos os resultados dos estudos realizados onde pudemos identificar problemas e questões no processo de produção e consumo de modelos de software triangulando dados da experiência de profissionais de desenvolvimento e dados de avaliação das ferramentas. Nesta triangulação identificamos achados das fases de investigação do contexto de desenvolvimento de software e investigação da interação com ferramentas de modelagem de software que apresentaram uma relação direta entre si, reforçando a validade desses achados. Outros achados que, mesmo sem uma relação direta, também apresentaram características relevantes que possibilitaram uma relação mediada entre questões relacionadas a ferramentas de modelagem e problemas identificados durante a investigação junto aos profissionais de desenvolvimento de software. Essas relações mediadas definiram espaços de investigação ainda não explorados, mas que já apresentam indicativos de impacto no processo de produção e consumo de modelos de software, considerando notações e ferramentas. Estes resultados nos apresentaram respostas para a questão de pesquisa "*Como ferramentas e notações de modelagem influenciam e apoiam o processo intelectual de produção e consumo de modelos de software através da interação humano-computador? "*, destacando os fatores TNP relacionados às formas de influência e apoio de ferramentas e notações no processo de produção e consumo de modelos de software.

6
Considerações Finais, Contribuições e Trabalhos Futuros

Esta pesquisa mostrou que as ferramentas e notações de modelagem podem causar diversos tipos de problemas de comunicabilidade e usabilidade para os profissionais envolvidos no processo de desenvolvimento de software, particularmente na etapa de modelagem deste processo. Tendo a tripla TNP como guia na investigação do contexto de desenvolvimento de software dos profissionais e na avaliação de ferramentas de modelagem de software, considerando sua comunicabilidade e usabilidade, pudemos caracterizar tais problemas e suas relações com ferramentas, notações e pessoas.

A relação entre problema de comunicabilidade e usabilidade de ferramentas de modelagem e problemas reportados por profissionais com experiência na prática do processo de desenvolvimento são respostas claras à nossa questão de pesquisa "*Como ferramentas e notações de modelagem influenciam e apoiam o processo intelectual da produção e consumo de modelos de software*". Na relação direta pudemos associar a forma como ferramentas de modelagem tratavam algumas situações (*i.e.* como o *designer* comunica e age sobre tal situação através da interface) com questões reportadas pelos profissionais sobre o processo de produção e consumo de modelos de software. Na relação mediada entre problemas identificados na investigação do contexto de desenvolvimento de software e outros identificados na investigação da interação com de ferramentas de modelagem de software pudemos apontar espaços para aprofundamento e extensão da investigação do processo de modelagem e de modelos de software em si. Um desses espaços seria o de investigação do contexto de desenvolvimento influenciando a modelagem de software, considerando pessoas, práticas realizadas e etc. Esta investigação já tem sido apontada como necessária em outras pesquisas recentes. (Petre, 2013) (Whittle et al. 2013) (Whittle et al., 2014)

A investigação do processo de produção e consumo de modelos, considerada como parte de um processo maior (i.e, o de desenvolvimento de software), nos possibilitou destacar o fator PESSOAS deste processo. Modelos de

software são construídos por e, normalmente, para pessoas numa sequência de atividades em que essas vão interagir com o modelo de diversas formas (ex. evoluir, transformar, publicar e etc.). Quando existe suporte computacional na produção e consumo de modelos de software, tendo em conta a Engenharia Semiótica como base conceitual, outro P deve ser considerado neste processo: o *designer* da ferramenta de modelagem. As evidências relacionadas à possiblidade de extensão das ferramentas de modelagem através de *plug-ins* indicam problemas e também um caminho de investigação complexo, onde qualquer usuário da ferramenta tem potencial para se tornar um *designer* desta. Sendo assim, a interação *designer*-usuário será uma conversa de 'n' *designer*s com o usuário. Sem uma articulação entre os discursos desses 'n' *designer*s a mensagem tem grande chance de ser confusa e problemática quando chega ao usuário. Esta seria uma perspectiva particularmente interessante para a comunidade de MDSE, que está considerando o *open-source* (vários *designer*s) como a única solução para modelagem na indústria desta abordagem *model-driven*. (Cabot, 2015a)

A caracterização do modelo como objeto de coordenação entre pessoas possibilitou a identificação de problemas relacionados ao uso, ou não, de objetos limítrofes (Star, 1990) nos modelos de software representando conceitos importantes do domínio do problema. Desenvolvedores de software devem conhecer o domínio do problema e seus conceitos de tal forma que possam fazer essa conexão entre problema e solução. Os modelos de software são meios de representação desses objetos em diferentes etapas do desenvolvimento. Outro tipo de problema relacionado ao modelo como objeto de coordenação está relacionado à notação UML, onde o estabelecimento de um protocolo social (Ellis, Gibbs, 1991) se mostrou necessário para dar suporte ao protocolo tecnológico provido por ferramentas de modelagem. A definição da porção da UML que será considerada num processo de modelagem não é da alçada da ferramenta. Considerando modelos como objeto de coordenação, também identificamos evidências da representação do modelo (Mnot) ser usada como um definidor de fronteiras do escopo do domínio e para representar fronteiras de atuação dos envolvidos no desenvolvimento.

Identificamos problemas relacionados com o objetivo dos modelos no desenvolvimento de software. A intenção de uso de modelos de software deve ser definida e de conhecimento de todos que produzam ou consomem de alguma

forma os modelos ao longo do processo. Alguns modelos de software são criados para uso prático e podem ser descartados assim que alcançam seu objetivo. Porém, outros modelos podem ter objetivos de mais longo prazo e, portanto, devem ser evoluídos e mantidos ao longo do desenvolvimento do software. Se os envolvidos na produção e consumo de modelos de software não tratarem estes de acordo com seus objetivos ou intenções, os reflexos podem chegar até o usuário final (ex. modelos desatualizados gerando sistema de ajuda do software).

A relação entre ferramentas e notação de modelagem possibilitou a identificação de diversos problemas para quem produz modelos de software. A validação da UML foi um desses problemas. As ferramentas não apresentam recursos confiáveis de validação da UML, o que pode causar problemas para todos os perfis de usuários. Mesmo os mais experientes podem desejar um suporte para verificação de um modelo ao final de sua construção, buscando identificar problemas simples como, por exemplo, um elemento sem transição de ou para outro elemento, ficando 'perdido' em um extenso modelo.

No cenário de produção e consumo de modelos de software investigado as evidências apontam que as ferramentas de modelagem têm foco na produção do modelo e não em seu consumo, o que pode causar diversos problemas na interação do consumidor.

Considerando o processo de produção e consumo de modelos de software como nosso contexto de investigação, foi possível observar que as ferramentas de modelagem não apoiam tal processo como um todo. Elas apresentam uma perspectiva mais focada no usuário (*user-centered design*). Na fase de investigação da interação com ferramentas de modelagem de software pudemos destacar características da metacomunicação de tais ferramentas mais relacionadas à produção de modelos, deixando o consumo como algo possível, porém não tão importante quando a produção. Existem vários recursos oferecidos pelas ferramentas para compartilhar modelos de software. Porém, a decisão sobre a forma de compartilhamento fica a cargo do produtor do modelo, podendo esta decisão dar indicações sobre a relação entre produtor e consumidor do modelo. Outra característica identificada é a mesma visão ser usada pelo produtor e consumidor do modelo. A "visão de produção" pode ser problemática durante a interação do consumidor com tais ferramentas. Além disso, se um modelo está sendo produzido para alguém consumir, seria interessante ter a visão do

consumidor durante a produção dos modelos. O produtor se colocar no lugar do consumidor durante a modelagem (consumindo o modelo através da ferramenta), dá oportunidade de corrigir problemas de modelagem ou realizar adaptações para ajudar no consumo de modelos (ex. Criar legendas no modelo, incluir anotações e etc.). Considerando a modelagem de software em seu contexto social, o produtor se colocar no lugar do consumidor pode ser uma perspectiva enriquecedora tanto para produtores de modelos quando para desenvolvedores de ferramentas de modelagem de software. Nas ferramentas avaliadas, o contexto social da modelagem não é efetivamente apoiado. A ferramenta Enterprise Architect oferece um modo *read-only* que pode ser entendido como uma funcionalidade de consumo dos modelos. Porém, pensando nesse cenário onde o produtor se coloca no lugar do consumidor, só é possível ver modelo e ferramenta como o consumidor se fechar o projeto e abrir o projeto *read-only* criado. A falta de flexibilidade para o produtor se colocar no lugar do consumidor indica que tal funcionalidade não tem nenhuma relação com esse objetivo e sim algo mais relacionado a "empacotar" o modelo em uma forma de consumo limitada (não editável).

A possibilidade de extensão da notação de representação de modelos nos apresentou evidências tanto em relação à extensão da notação UML (ex. uso de estereótipos, *tags e constraints* mecanismos de extensão definidos pela UML) quanto da notação de modelagem como um todo (ex. recursos de desenho ou alteração de representação de elementos UML). A flexibilidade na produção do modelo é uma necessidade indicada pelos profissionais de desenvolvimento. Porém, esta flexibilidade demanda uma maior responsabilidade do produtor do modelo quando está se apropriando da notação de modelagem (Mnot) para construir a mensagem do modelo (Msem). A produção deste discurso deve considerar a forma, o conteúdo do modelo e também o contexto em que o modelo será consumido. Se produtor e consumidor compartilharem o contexto de produção do modelo, os problemas de comunicação dos modelos podem ser identificados e corrigidos (força de coerção do modelo) em tempo de consumo do modelo. Porém, se o contexto não é compartilhado o modelo tem a missão de expressar e reconstituir o contexto para o consumidor. Neste último caso, a representação do modelo (Mnot) tem grande impacto no significado (Msem) que este modelo irá comunicar, assim como a ferramenta usada para tal produção.

A produção do modelo caracterizada como uma produção de discurso do produtor para o consumidor nos apontou questões evidenciadas nos achados sobre o papel dos modelos na comunicação entre pessoas. Nesta pesquisa identificamos junto aos profissionais o uso de ferramentas de modelagem, a natureza de ferramenta também avaliada nesta pesquisa; e também o uso de 'desenhadores' de modelos. As ferramentas de modelagem avaliadas também oferecem recursos de desenho. Entretanto, não foi possível identificar nenhum recurso considerando o modelo como um instrumento de comunicação entre pessoas. A comunicação através dos modelos tem relação com o papel epistêmico dos modelos de software durante o processo de desenvolvimento. A representação do modelo (Mnot) é uma abstração feita pelo produtor do que este achou relevante comunicar (Msem) considerando a intenção do modelo produzido e seu consumidor. As escolhas do produtor do modelo em relação a notações e conteúdo podem ser influenciadas pelo suporte que a ferramenta oferece. As ferramentas de modelagem estão olhando para 'como desenhar' e/ou 'como modelar', mas não para 'como comunicar', deixando essa responsabilidade para o produtor do modelo.

Alguns *insights* foram identificados nesta pesquisa sobre a questão no longo prazo sobre a relação das experiências de IHC de desenvolvedores durante o desenvolvimento de software e as experiências de IHC dos usuários finais desse software desenvolvido. Porém, não podemos fazer nenhuma afirmação sobre esta questão, já que concentramos nossa investigação somente no contexto de modelagem de software, que é somente uma parte do processo de desenvolvimento de software. Os resultados desta pesquisa podem ser usados como insumo para a evolução de investigações e discussões sobre propagação de problemas de IHC de desenvolvedores de software para o usuário final em pesquisas futuras. O relato de um dos participantes dos estudos reforçou nossa crença na questão de pesquisa no longo com uma evidência interessante sobre como os modelos produzidos e consumidos ao longo do processo de desenvolvimento podem influenciar o software produzido:

"... das classes, pra verificar se a gente não ter perdas, porque não adianta você fazer um bom trabalho em negócio e requisitos, e quando você começa a construir o software mesmo, fazer a tradução em objetos, ***se você tiver perdas, se você não tiver a mesma linguagem, procurar ter as mesmas associações, você***

vai construir um software que não está adequando, que não está aderente ao negócio*...” (Participante 7)*

Esta pesquisa apresenta como limitações: 1) investigação do processo de modelagem de software considerando somente a UML, e somente uma porção desta, como notação; 2) a análise e categorização dos dados foram executadas somente pela pesquisadora; e 3) Não foi realizado um estudo empírico com modeladores interagindo com ferramentas de modelagem.

Sobre a limitação (1), apesar da UML ser a notação de modelagem mais conhecida e utilizada no desenvolvimento de software, existem outras notações neste contexto, inclusive presentes nos achados tanto da fase de investigação do contexto de desenvolvimento de software quanto da fase de interação com ferramenta. Portanto, nossos achados só tratam de parte das notações existentes e utilizadas para modelagem de software.

Sobre a limitação (2) esta está relacionada à metodologia da pesquisa onde somente uma pessoa (a pesquisadora) realizou a análise e categorização dos dados coletados nos estudos. O objetivo principal desta pesquisa era investigar em profundidade como ferramentas e notações de modelagem influenciam e apoiam o processo intelectual da produção e consumo de modelos de software, sem a intenção de buscar padrões ou generalidades. Portanto, a decisão de ter somente uma pessoa analisando os dados foi consciente buscando a caracterização de questões ainda não conhecidas. Mesmo conscientes dessa decisão, entendemos que ter mais de uma pessoa analisando os mesmos dados pode produzir resultados mais ricos e a ocorrência de um viés de pesquisa pode ser minimizado.

Sobre a limitação (3), a decisão de não realizar estudos empíricos com modeladores interagindo com ferramentas de modelagem está relacionado a profundidade dos achados no processo de inspeção das ferramentas. Então, para uma triangulação com situações de interação de usuários com as ferramentas que contemplassem todo o espectro de achados, seria necessário realizar estudos longitudinais, perseguindo as possíveis consequências dos problemas de usabilidade e comunicabilidade desde o momento da modelagem até o momento da interação com o usuário final. Ou seja, seria necessário investigar a conjectura de propagação em um caso real de desenvolvimento. Qualquer triangulação com

fragmentos deste percurso seria uma evidência, mas não suficientemente forte para validar por si só toda a extensão de achados obtidos. Assim, esta é uma limitação relacionada ao contexto do tempo de elaboração e às condições de conclusão de uma tese de doutorado, e que deixa como trabalhos futuros a realização destas pesquisas que poderão confirmar ou desconfirmar os achados que você apresenta (e as relações que eles têm uns com outros e, todos em conjunto, com pesquisas de terceiros).

Além das contribuições para o contexto de modelagem de software em si, com resultados e achados caracterizados através da tripla TNP, esta pesquisa também apresentou contribuições para evolução da teoria das IHC e, particularmente, da Engenharia Semiótica.

6.1
Contribuições para pesquisa de IHC

Para a área de IHC, a contribuição desta pesquisa foi a combinação das perspectivas semiótica e cognitiva para avaliação de ferramentas de modelagem. Cada perspectiva tem o seu foco e profundidade de investigação, e com a associação das duas perspectivas através do desenvolvimento do *Combined Semiotic-Cognitive Evaluation method* (CSCE) buscamos ampliar nosso escopo de avaliação, dando conta de questões de comunicabilidade e usabilidade, relacionando uma à outra. Além das perspectivas combinadas, a tripla TNP, outra contribuição desta pesquisa, foi apresentada como um recurso de articulação do contexto de modelagem de software, específico desta pesquisa. Entendemos que o CSCE associado à tripla TNP ainda apresentam como resultado uma visão parcial do contexto investigado, e, portanto das questões e problemas. O CSCE, assim como os métodos combinados para sua construção, tem como foco de avaliação uma porção da ferramenta considerando um determinado cenário e perfil de usuário, definidos pela base da tripla TNP. Essa visão focada é bastante interessante para o aprofundamento dos problemas e questões identificados, mas não oferece uma visão da comunicabilidade e usabilidade da ferramenta como um todo. Porém, a visão combinada proposta é mais ampla do que as visões das duas perspectivas isoladamente. A dinâmica de experimentação variando o ponto de observação do fenômeno, ora mais focado, ora mais abrangente (*zoom-in/zoom-out*) da pesquisa também se mostrou como uma contribuição para a avaliação de

IHC, ampliando o escopo de avaliação, possibilitando uma visão mais rica do fenômeno da interação humano-computador.

6.2
Contribuições para Engenharia Semiótica

Como contribuição para a Engenharia Semiótica, apresentamos uma nova perspectiva sobre o processo de desenvolvimento de software que resulta, em última análise, na experiência do usuário durante a interação com software. Conhecer melhor as condições de comunicação e mútuo entendimento daqueles que, segundo a teoria, 'comunicam sua intenção e princípios de *design* através da interface' é uma fonte valiosa de explicações sobre problemas de comunicabilidade em IHC.

Considerando o contexto de produção e consumo do modelo, também indicamos o potencial de extensão o conceito de comunicabilidade usado na Engenharia Semiótica, transferindo essa característica, originalmente proposta para o âmbito da metacomunicação *designer*-usuário, para o produto ou artefato gerado desta metacomunicação, que no contexto desta pesquisa foram os modelos de software. No contexto desta pesquisa, observamos a comunicação *designer*-produtor do modelo e *designer*-consumidor do modelo, considerando o conceito original da comunicabilidade, mas também observamos os modelos produzidos no processo de produção e consumo de modelos como mensagens do produtor ao consumidor do modelo. (Figura 57) O modelo de software como preposto de seu produtor. A comunicabilidade dos modelos não foi o foco desta pesquisa, porém alguns achados podem ser usados para pesquisas mais específicas nesta linha, quando os modelos são produzidos e consumidos através de uma ferramenta de modelagem.

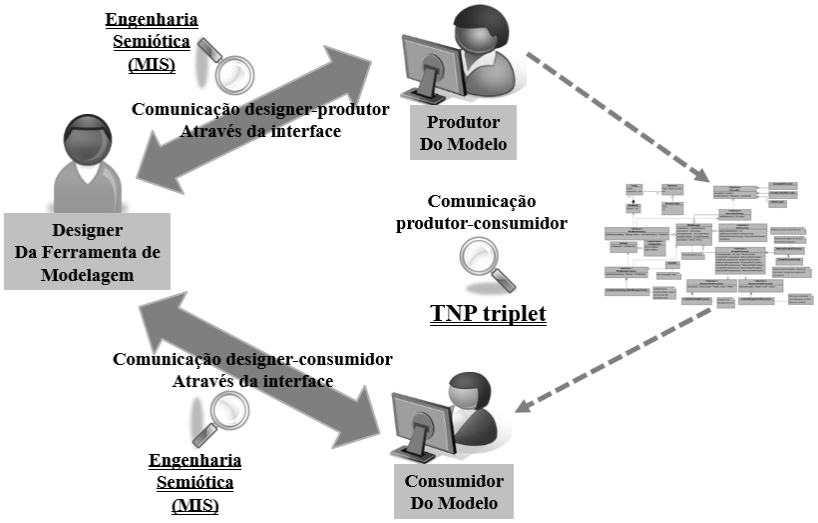


Figura 57. Os "Ps" investigados

6.3
Trabalhos Futuros

O Método de Avaliação combinado Semiótico cognitivo (CSCE) associado com a tripla TNP só foi aplicado, na íntegra, no contexto desta pesquisa. Apesar do método não ser um método de fácil execução, sua aplicação em outros contextos relevantes, no próprio contexto de desenvolvimento de software, pode ser interessante para a evolução do método e para o contexto de desenvolvimento de software, caracterizando outros cenários e outras notações de produção de artefatos intelectuais no processo de construção do software.

Associado a uma das limitações da pesquisa, temos como trabalho futuro a realização de estudos longitudinais, perseguindo as possíveis consequências dos problemas de usabilidade e comunicabilidade desde o momento da modelagem até o momento da interação com o usuário final. Estas pesquisas poderão confirmar ou desconfirmar os achados apresentados nesta pesquisa (e as relações que eles têm uns com outros e, todos em conjunto, com pesquisas de terceiros).

Algumas categorias de questões identificadas junto aos profissionais da fase de investigação do contexto de desenvolvimento de software (ex. Abandono de Modelos, Modelos úteis), até onde esta pesquisa foi executada, não puderam ser associadas diretamente a experiências de IHC. A aplicação da investigação para outros tipos ou natureza de ferramentas, com características mais ubíquas, entrando mais nos processos sociais, podem enriquecer os resultados relacionados às questões aqui identificadas.

A necessidade de flexibilidade das ferramentas de modelagem de software foi indicada em pesquisas anteriores. (Petre, 2013) (Petre, 2014) (Whittle et al, 2013) Porém, pouco se sabe sobre o que seria essa flexibilidade. Para suporte ao processo de *design rationale*, comum das fases iniciais do desenvolvimento de software (Carrol, 2012) (Petre, Van der Hoek, 2013), as ferramentas precisam fazer a gestão de conhecimento desse *design rationale*, para apoiar os desenvolvedores. Esse contexto em particular seria um potencial contexto de investigação, onde o CSCE e a tripla TNP poderiam ajudar na caracterização dos problemas e, consequentemente, na identificação das características de flexibilidade necessárias para apoiar os estágios iniciais do *design* de software.

Por último, a faceta de coordenação dos modelos não foi investigada no contexto desta pesquisa, considerando a perspectiva da interação da ferramenta de modelagem. Porém, considerando os achados na investigação do contexto de desenvolvimento de software, se mostra como um espaço rico de pesquisa, potencialmente relacionando as áreas de IHC e de CSCW ("*Computer Supported Cooperative Work*").

7
Referências

AFONSO, L. M.; CERQUEIRA, R.F.G; DE SOUZA, C.S. Evaluating application programming interfaces as communication artefacts. **In Proceedings of the Psychology of Programming Interest Group Annual Conference 2012** PPIG'2012 November 21-23,2012 - London, UK. pp. 151-162, 2012.

AMBLER, S. W. **Agile modeling: effective practices for extreme programming and the unified process**. Wiley.com, 2002.

ARIAS, E., FISCHER, G. Boundary Objects: Their Role in Articulating the Task at Hand and Making Information Relevant to It. **In Proceedings of the International ICSC Symposium on Interactive & Collaborative Computing (ICC'2000),** University of Wollongong, Australia, ICSC Academic Press, Wetaskiwin, Canada, pp. 567-574. 2000.

BARBOSA, C. M. A. **Manas: uma ferramenta epistêmica de apoio ao projeto da comunicação em sistemas colaborativos.** Pontifícia Universidade Católica do Rio de Janeiro-PUC-Rio, 2006.

BARBOSA, S. D. J. SILVA, B. S. **Interação humano-computador**. Elsevier. 2010.

BLACKWELL, A. F. Introduction Thinking with Diagrams. In: **Thinking with Diagrams**. Springer Netherlands, p. 1-3. 2001.

BLACKWELL, A., GREEN, T. Notational systems: The cognitive dimensions of notations framework **In J. M. Carroll (Ed.), HCI models, theories and frameworks: Toward a multidisciplinary science**. San Francisco, 103– 134. 2003.

BLANDFORD, A. Semi-structured qualitative studies. **In: Soegaard, Mads and Dam, Rikke Friis (eds.). The Encyclopedia of Human-Computer Interaction**, 2nd Ed.. Aarhus, Denmark: The Interaction Design Foundation. 2014. Disponível online:https://www.interaction-design.org/encyclopedia/semi-structured_qualitative_studies.html

BRAMBILLA, M.; CABOT, J.; WIMMER, M.. Model-driven software engineering in practice. **Synthesis Lectures on Software Engineering**, v. 1, n. 1, p. 1-182, 2012.

BRUNELIERE, H., CABOT, J. On Developing Open Source MDE Tools: Our Eclipse Stories and Lessons Learned. **In: Open Source Software for Model Driven Engineering (OSS4MDE'14) workshop**-a MODELS 2014 Satellite Event. 2014.

BUDGEN, D. The Cobbler's Children: Why Do Software Design Environments Not Support Design Practices? **Software *Designer*s in Action: A Human-Centric Look at Design Work**. Abingdon: Chapman and Hall/CRC, 199-215. 2013.

BUDGEN, D., BURN, A. J., BRERETON, O. P., KITCHENHAM, B. A., PRETORIUS, R. Empirical evidence about the UML: a systematic literature review. **Software: Practice and Experience**, v. 41, n. 4, p. 363-392, 2011.

CABOT J. Modeling Languages [blog na internet]. Jordi Cabot. **Drawing tools vs modeling tools**. Abril 2011 [citado em Fevereiro de 2015]. Disponível em: http://modeling-languages.com/drawing-tools-vs-modeling-tools/. 2011.

CABOT J. Modeling Languages [blog na internet]. Jordi Cabot. **Modeling in industry: open source is the only solution**. Janeiro 2015 [citado em Fevereiro de 2015]. Disponível em: http://modeling-languages.com/modeling-open-source-industry/. 2015a.

CABOT J. Modeling Languages [blog na internet]. Jordi Cabot. **Poor validation of UML Models in Eclipse UML2 tools**. Abril 2010[citado em Fevereiro de 2015]. Disponível em: http://modeling-languages.com/poor-validation-uml-models-eclipse-uml2-tools/?utm_source=twitter&utm_medium=evergreen_post_tweeter&utm_campaign=website_. 2010.

CABOT J. Modeling Languages [blog na internet]. Jordi Cabot. **UML tools**. Fevereiro 2015 [citado em Fevereiro de 2015]. Disponível em: http://modeling-languages.com/uml-tools/. 2015b.

CANOE, C. H.; ROSSON, M. Articulating collaborative contributions to activity awareness. **Theories of team cognition: Cross-disciplinary perspectives**, p. 209, 2012.

CARROLL, J. M. Creativity and Rationale: The Essential Tension. **Creativity and Rationale: Enhancing Human Experience by Design.** Springer Science & Business Media, p. 1-10. 2012.

CARROLL, J. M., ROSSON, M. B., Farooq, U., Xiao, L. Beyond being aware. **Information and Organization**, 19(3), 162-185. 2009.

CHOMSKY, N. **Linguagem e pensamento**. Petrópolis: Vozes, 1971.

CLARKE, S. Evaluating a new programming language. **In: Proceeding of 13th Workshop of the Psychology of Programming Interest Group**. p. 275-289. 2001.

CRESWELL, J.W. **Research Design: Qualitative, Quantitative, and Mixed Methods Approaches**. Sage Publications. 2013.

CUADRADO, J. S., GUERRA, E., DE LARA, J. Generic model transformations: write once, reuse everywhere. **In: Theory and Practice of Model Transformations**. Springer Berlin Heidelberg, 2011. p. 62-77. 2011.

DE SOUZA, C. R., QUIRK, S., TRAINER, E., REDMILES, D. F. Supporting collaborative software development through the visualization of socio-technical dependencies. **In: Proceedings of the 2007 international ACM conference on Supporting group work**. ACM, 2007. p. 147-156. 2007.

DE SOUZA, C. R.; FROEHLICH, J.; DOURISH, P. Seeking the source: software source code as a social and technical artifact. **In:Proceedings of the 2005 international ACM SIGGROUP conference on Supporting group work**. ACM, 2005. p. 197-206. 2005.

DE SOUZA, C. S. **The Semiotic Engineering of Human–Computer Interaction**. Cambridge, MA. The MIT Press. 2005.

DE SOUZA, C. S., LEITÃO, C. F. **Semiotic engineering methods for scientific research in HCI.** Princeton: NJ. Morgan & Claypool. 2009.

DE SOUZA, C.S.; LEITÃO, C.F.; PRATES, R.O.; BIM, S.A.; DA SILVA, E.J. Can inspection methods generate valid new knowledge in HCI? The case of semiotic inspection. **International Journal of Human-Computer Studies**, Volume 68, Issues 1-2, January-February 2010, Pages 22-40. 2010.

DOBING, B. PARSONS, J. How UML is used. **Communications of the ACM**, v. 49, n. 5, p. 109-113, 2006.

EASTERBROOK, S., SINGER, J. Selecting empirical methods for software engineering research. **Guide to Advanced Empirical Software Engineering**, pp. 285–311. 2008.

ECO, U. **A Theory of Semiotics**. Bloomington: Indiana University Press, 1976.

ECO, U. **Semiotics and the Philosophy of language**. Bloomington: Indiana University Press, 1983.

ECO, U. **Tratado geral de semiótica**. Tradução de Antônio de Pádua Danesi e Gilson Cesar Cardoso de Souza. Ed. Perspectiva. São Paulo. 2000.

ELLIS, C., GIBBS, S. Groupware: some issues and experiences. **Communications of the ACM**, 34, 1991.

ERICKSON, J.; SIAU, K.. Theoretical and practical complexity of modeling methods. **Communications of the ACM**, v. 50, n. 8, p. 46-51, 2007.

EVANS, E. **Domain-driven design: tackling complexity in the heart of software**. Addison-Wesley Professional, 2004.

FERREIRA, J. J., DE SOUZA, C. S., CERQUEIRA, R. Characterizing the tool-notation-people Triplet in software modeling tasks. In **Proceedings of the 13th Brazilian Symposium on Human Factors in Computing Systems (IHC'14)**. Sociedade Brasileira de Computação, Porto Alegre, Brazil, Brazil, 31-40. 2014.

FERREIRA, J.J.; DE SOUZA, C. S. Communicating ideas in computer-supported modeling tasks: A case study with BPMN. **In Human-Computer Interaction.**

Human-Centred Design Approaches, Methods, Tools, and Environments, Springer Berlin Heidelberg, 320-329. 2013.

FERREIRA, J.J.; DE SOUZA, C.S.; SALGADO, L.C.C.; SLAVIEIRO, C.; LEITÃO, C.F; MOREIRA, F.F.: Combining cognitive, semiotic and discourse analysis to explore the power of notations in visual programming. **In Proceedings of VL-HCC'2012**, 101-108, 2012.

FRANCE, R., RUMPE, B**.** Model-driven development of complex software: A research roadmap. **Future of Software Engineering**. IEEE Computer Society, 2007.

FUNT, B. V. Problem-Solving with Diagrammatic Representation. **Diagrammatic reasoning: Cognitive and computational perspectives**. MIT Press, p. 33-68. 1995.

GEE, J. P. **An Introduction to Discourse Analysis: Theory and Method**. London: Routledge. 2005.

GREEN, T.; BLACKWELL. A. Cognitive dimensions of information artifacts: a tutorial. **In Proceeding of BCS HCI Conference**. 1998.

INGRAM, D. **First Language Acquisition: Method, Description and Explanation** Cambridge University Press, 1989.

INGRAM, D. The relationship between comprehension and production. R. L. Schiefelbusch, L. L. Lloyd (eds.). **Language Perspectives: Aquisition, Retardation, and Intervention**. Baltimore: Universitu Park Press, pp. 313-34. 1974

JACOBSON, I.. A Resounding 'Yes' to Agile Processes—But Also More. **Cutter IT Journal**, vol. 15, no. 1, January 2002, pp. 18–24. 2002.

JAKOBSON, R. Linguistics and poetics. **In: Sebeok, T.A.(Ed.) Style in language.** Cambridge, MA: The MIT Press, p. 350-377, 1960.

KENT, S. Model driven engineering. **Integrated Formal Methods**. Springer Berlin Heidelberg, 2002.

KIRSH, D., MAGLIO, P. On distinguishing epistemic from pragmatic action. **Cognitive science**, v. 18, n. 4, p. 513-549, 1994.

KÜHNE, T. Matters of (meta-) modeling. **Software & Systems Modeling** 5.4: 369-385. 2006.

KUTAR, M., BRITTON, C., BARKER, T. A comparison of empirical study and cognitive dimensions analysis in the evaluation of UML diagrams. **In: Proceeding of the 14th Workshop of the Psychology of Programming Interest Group (PPIG 14).** 2002.

LANO, K., KOLAHDOUZ-RAHIMI, S. Model-driven development of model transformations. **In: Theory and practice of model transformations**. Springer Berlin Heidelberg, p. 47-61. 2011.

LARKIN, J. H., SIMON, H. A. Why a Diagram is (sometimes) worth then thousand words. **Diagrammatic reasoning: Cognitive and computational perspectives**. MIT Press, p. 69-109. 1995.

LAZAR, J., FENG. J. H., HOCHHEISER, H. **Research Methods in Human-Computer Interaction**. Wiley Publishing. 2010.

LEE, C. P. **Between chaos and routine: boundary negotiating artifacts in collaboration**. In Proceedings of the Ninth European Conference on Computer-Supported Cooperative Work, Paris, France. Springer Netherlands, p. 387-406. 2005.

LEITÃO, C. F., SILVEIRA, M. S., DE SOUZA, C. S. Uma introdução à engenharia semiótica: conceitos e métodos. **In Proceedings of the 12th Brazilian Symposium on Human Factors in Computing Systems (IHC '13)**. Brazilian Computer Society, Porto Alegre, Brazil, Brazil, 356-358. 2013.

LEITÃO, C.F. Métodos Qualitativos em Pesquisa Científica. **Computação Brasil** (no. 11, Out/Dez 2009). Porto Alegre: Sociedade Brasileira de Computação. p. 22-23. 2009.

LIEBERMAN, B. A. **The art of software modeling**. CRC Press, 2006.

LUDEWIG, J. Models in software engineering–an introduction. **Software and Systems Modeling**, v. 2, n. 1, p. 5-14, 2003.

MAGNANI, L. An abductive theory of scientific reasoning. **Semiotica**, v. 2005, n. 153-1/4, p. 261-286, 2005.

MARAEE, A.; BALABAN, M. Removing Redundancies and Deducing Equivalences in UML Class Diagrams. **Model-Driven Engineering Languages and Systems**. Springer International Publishing, p. 235-251. 2014.

MCCALL, R. Critical conversations: Feedback as a stimulus to creativity in software design. **Creativity and Rationale.** Springer London. p. 11-40. 2013

MELLOR, S. J., CLARK, A. N., FUTAGAMI, T.. Model-driven development:. guest editors' introduction. **IEEE software**, v. 20, n. 5, p. 14-18. 2003.

MONTEIRO, I.T.; DE SOUZA, C.S.; LEITÃO, C.F. Metacommunication and semiotic engineering: Insights from a study with mediated HCI. **In: HCI International 2013 - Design, User Experience, and Usability. Design Philosophy, Methods, and Tools.** 2013, Las Vegas. LNCS Proceedings of HCI International. Heidelberg: Springer, 2013. vol. 8012. pp. 115-124. 2013.

MOODY, D. L. The "physics" of notations: toward a scientific basis for constructing visual notations in software engineering. **Software Engineering**, IEEE Transactions on, v. 35, n. 6, p. 756-779, 2009.

MORIYON, R., SZEKELY, P., NECHES, R. Automatic Generation of Help from Interface Design Models. **In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems**. ACM, p. 225-231. 1994.

MULLER, P. A., FONDEMENT, F., BAUDRY, B., COMBEMALE, B. Modeling modeling modeling. **Software & Systems Modeling**, 11(3), 347-359. 2012.

MUSSBACHER, G., AMYOT, D., BREU, R., BRUEL, J. M., CHENG, B. H., COLLET, P., COMBEMALE, B., FRANCE, R. B., HELDAL, R., HILL, J., KIENZLE, J., SCHÖTTLE, M., STEIMANN, F., STIKKOLORUM, D., WHITTLE, J. The Relevance of Model-Driven Engineering Thirty Years from Now. **In: Model-Driven Engineering Languages and Systems**. Springer International Publishing. p. 183-200. 2014.

MYERS, B., PARK, S. Y., NAKANO, Y., MUELLER, G., KO, A., How *designer*s design and program interactive behaviors. **In: Visual Languages and Human-Centric Computing**, 2008. VL/HCC 2008. IEEE Symposium on. IEEE. p. 177-184. 2008.

NORMAN, D. A. (Ed.). **Perspectives on cognitive science**. USA: Ablex Publishing Corporation, 1981.

NORMAN, D. A.: Cognitive engineering. In **D. A. Norman, S. W. Draper. *User*-centered system design**. Hillsdale, NJ: Laurence Erlbaum. 31–62, 1986.

OMG, **Unified Modeling Language (UML), V2.4.1 - Superstructure**. Object Management Group (OMG), OMG Document formal/2011-08-05, 2011. Acessado em Fevereiro de 2015.

PANGOLI, S.; PATERNÒ, F. Automatic Generation of Task-Oriented Help. **In Proceedings of the 8th annual ACM symposium on User interface and software technology**. ACM, 1995. p. 181-187. 1995.

PEIRCE, C. S**. The essential Peirce: Selected Philosophical Writings**. Vols. I , II. N. Houser and C. J. W. Kloesel (Eds.). Bloomington, IN. Indiana University Press, 1992-1998.

PETRE, M. Reflections on Representations: Cognitive Dimensions Analysis of Whiteboard Design Notations. **Software *Designer*s in Action: A Human-Centric Look at Design Work.** Abingdon: Chapman and Hall/CRC. 199-215. 2013a.

PETRE, M. UML in practice. **In: Proceedings of the 2013 International Conference on Software Engineering**. IEEE Press. p. 722-731. 2013b.

PETRE, M. Why looking isn't always seeing: readership skills and graphical programming. **Communications of the ACM**, v. 38, n. 6, p. 33-44, 1995.

PETRE, M., VAN DER HOEK, A. (Ed.). **Software *Designer*s in Action: A Human-centric Look at Design Work**. CRC Press, 2013.

PETRE, M.. “No shit” or “Oh, shit!”: responses to observations on the use of UML in professional practice. **Software & Systems Modeling**, v. 13, n. 4, p. 1225-1235, 2014.

PIMENTEL, M.; FUKS, H. (organizadores). **Sistemas colaborativos**. Rio deJaneiro–RJ: Elsevier-Campus-SBC, 2011.

PRATES, R. O. **Engenharia Semiótica de Linguagens de Interfaces Multi-Usuário**. Tese de doutorado, Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, Brasil, 1998.

PRATES, R. O.; DE SOUZA, C. S.; BARBOSA, S. D. J. A method for evaluating the communicability of user interfaces. **Interactions.** 7(1): 31-38. 2000.

PRESSMAN, R. S**. Software engineering : a practitioner’s approach** - 7th ed. New York: McGraw-hill, 2010.

REGGIO, G.; LEOTTA, M.; RICCA, F. Who Knows/Uses What of the UML: A Personal Opinion Survey. **In: Model-Driven Engineering Languages and Systems**. Springer International Publishing. p. 149-165. 2014.

RHEINFRANK, J.; EVENSON, S. Design languages. **In: Bringing design to software.** ACM. p. 63-85. 1996.

RUMBAUGH, J., JACOBSON, I., BOOCH, G. **The UML reference manual**. New York: Addison-Wesley, v. 1, p. 999, 1999.

RUMPE, B. Agile modeling with the UML. **In: Radical Innovations of Software and Systems Engineering in the Future**. Springer Berlin Heidelberg, p. 297-309, 2004.

SAMPAIO, A. L. **Um modelo para descrever e negociar modificações em sistemas Web**. Pontifícia Universidade Católica do Rio de Janeiro-PUC-Rio, 2010.

SANTAELLA, L. Abduction: The logic of guessing. **Semiotica**, v. 2005, n. 153-1/4, p. 175-198, 2005.

SANTAELLA, L. **O Método Anti-Cartesiano de C. S. Peirce**. UNESP/FAPESP. 2004.

SCHÖN, D. A. **The reflective practitioner: How professionals think in action**. Basic books, 1983.

SCHÖN, D., BENNETT, J. Reflective conversation with materials. **In: Bringing design to software.** ACM. p. 171-189. 1996.

SELIC, B. The pragmatics of model-driven development. **IEEE Software**, 20(5), 19–25. doi:10.1109/MS.2003.1231146.2003.

SOMMERVILLE, I. **Software Engineering**. 9th edition. Harlow, England: Addison-Wesley, 2010.

STACHOWIAK, H.:**Allgemeine Modelltheorie**.Springer,Wien. 1973.

STAR, S. L. The structure of ill-structured solutions: boundary objects and heterogeneous distributed problem solving, **Morgan Kaufmann Series In Research Notes In Artificial Intelligence** , 37—54. 1990.

STAR, S. L.. This is not a boundary object: Reflections on the origin of a concept. **Science, Technology & Human Values**, v. 35, n. 5, p. 601-617, 2010.

STENNING, K., OBERLANDER, J. A cognitive theory of graphical and linguistic reasoning: Logic and implementation. **Cognitive science**, v. 19, n. 1, p. 97-140, 1995.

SUTCLIFFE, A. Integrating Design Representations for Creativity. **In:Creativity and Rationale**. Springer London. p. 85-104. 2013.

WHITEHEAD, J. Collaboration in software engineering: A roadmap. **In:2007 Future of Software Engineering**. IEEE Computer Society, p. 214-225. 2007.

WHITTLE, J., HUTCHINSON, J., ROUNCEFIELD, M. The State of Practice in Model-Driven Engineering. **Software**, IEEE, v. 31, n. 3, p. 79-85, 2014.

WHITTLE, J., HUTCHINSON, J., ROUNCEFIELD, M., BURDEN, H., HELDAL, R. Industrial adoption of model-driven engineering: are the tools really the problem?. **In: Model-Driven Engineering Languages and Systems**. Springer Berlin Heidelberg, p. 1-17. 2013.

WINOGRAD, T.(Ed.). **Introduction- *Bringing Design to Software***. *Bringing Design to Software*. ACM, New York, NY, USA. 1996.

ZHENG, Y., TAYLOR, R. N. A classification and rationalization of model-based software development. **Software & Systems Modeling**, v. 12, n. 4, p. 669-678, 2013.

8
Apêndices

8.1
Apêndice A

QUESTIONÁRIO E TERMO DE CONSENTIMENTO RELACIONADO

LEVANTAMENTO SOBRE EXPERIÊNCIA EM DESENVOLVIMENTO DE SOFTWARE

O termo de consentimento e questionário relacionado tem o objetivo de levantar informações sobre experiências e conhecimento de profissionais que atuaram ou atuam em projeto de Desenvolvimento de Software, particularmente experiências relacionadas à construção e uso de modelos durante o processo de desenvolvimento de software.

TERMO DE CONSENTIMENTO PARA APLICAÇÂO DE QUESTIONÁRIO

Título da pesquisa: Comunicação através de modelos no contexto de desenvolvimento de Software

Pesquisadora: Juliana Jansen Ferreira (Programa de Doutorado em Informática, PUC-Rio)

Caro participante, sou aluna do Programa de Pós-Graduação em Informática da PUC-Rio, tendo como orientadora a Professora Clarisse Sieckenius de Souza, e gostaria de contar com sua participação na minha pesquisa de doutorado.

Sobre a pesquisa:

Minha pesquisa tem por objetivo investigar a utilização de modelos no desenvolvimento de software como objetos de comunicação entre a equipe envolvida no processo de desenvolvimento. A contribuição esperada desta pesquisa é o aprimoramento dos processos de comunicação ao longo do ciclo de desenvolvimento de software, com o uso de modelos já construídos e utilizados durante este processo. A proposta é apresentar oportunidades de adequação de ferramentas, indicações sobre escolha e uso de notações e modelos e procedimentos de uso de tais modelos, de forma que o processo de comunicação entre os envolvidos no desenvolvimento de software tenha seus ruídos minimizados. Com menos falhas de comunicação durante o processo de desenvolvimento, várias das situações de retrabalho e as inconsistências de artefatos produzidos podem ser prevenidas e causar assim menor impacto na construção do software.

Este estudo é composto por um questionário que tem como **objetivo identificar o perfil do participante e sua experiência com modelos de software ao longo de sua atuação em projetos de desenvolvimento de software**. Para este estudo, solicito seu consentimento para a participação voluntária, tendo a **duração aproximada de 20 minutos**. O questionário é composto por **15 perguntas**. Para a maioria delas, são apresentadas opções de resposta, ou seja, o participante só precisa marcar uma ou mais opções. Porém, em alguns casos, convidamos o participante a comentar sua resposta, buscando dados mais ricos para a pesquisa. Fique à vontade para indicar suas respostas e adicionar comentários, ou somente marcar a resposta sem comentá-la. Agradeço sua disposição em colaborar com a minha pesquisa da maneira que se sentir mais confortável. Para que você decida se quer colaborar, é importante fornecer algumas informações gerais:

1) Os **tipos de dados coletados** são:

   i. Respostas relacionadas ao perfil do participante em relação a sua experiência em projeto de desenvolvimento de software
   ii. Respostas sobre experiências, comentários e relatos do participante sobre construção e uso de modelos de software;

2) Toda pesquisa pauta-se nas as **normas éticas de conduta em pes**quisa, que garantem o **respeito à privacidade e ao anonimato** dos participantes que consentirem em participar. Também respeitando tais normas, os dados coletados destinam-se **estritamente** a atividades de investigação científica e desenvolvimento técnico, e são vistos e usados **exclusivamente** pela equipe de pesquisadores envolvida na pesquisa em questão. Esses dados podem ser solicitados pelo respectivo participante a qualquer momento da pesquisa. Além disto, ao divulgar os resultados do estudo em foros científicos, me comprometo rigorosamente a manter **sigilo** sobre quaisquer dados que ponham em risco o anonimato dos colaboradores e das empresas envolvidas.

3) A pesquisa não oferece nenhum risco ao participante e o consentimento para colaborar é uma escolha livre, que pode ser revisto sempre que o participante assim desejar. O participante pode desistir de colaborar com a pesquisa a qualquer momento sem que lhe traga qualquer prejuízo ou constrangimento. Eu, Juliana Jansen Ferreira, estou, portanto, disponível para prestar esclarecimentos e tirar quaisquer dúvidas sobre os procedimentos de pesquisa durante todo o período de sua execução. Para contato, fale diretamente comigo durante as atividades do projeto, ligue para o telefone 99432-5470, ou se preferir use o e-mail *jferreira@inf.puc-*

rio.br. Para contatar a professora Clarisse Sieckenius de Souza, ligue para 3527-1500 ramal 4344, ou se preferir use o e-mail *clarisse@inf.puc-rio.br*.

4) Fique à vontade para imprimir este termo para futuras referências ou esclarecimentos. Caso deseje uma cópia assinada pelos pesquisadores, favor enviar um email para jferreira@inf.puc-rio.br solicitando.

De posse das informações acima, pediria então que você, por favor, se pronunciasse acerca de seu consentimento para coleta de dados durante as etapas do estudo, marcando uma das opções abaixo e depois adicionando seu nome e assinatura.

☐ Li e entendi as informações neste termo

☐ Autorizo a coleta e uso de dados para pesquisa
☐ Não autorizo a coleta e uso de dados para pesquisa

☐ Gostaria de mais informações sobre a coleta e uso de dados

Nome do participante:

Email do participante:

Independentemente de sua decisão, agradecemos a atenção que nos dispensa.

Juliana Jansen Ferreira

Doutoranda do Programa de Pós-Graduação em Informática do Departamento de Informática da PUC-Rio

QUESTIONÁRIO

LEVANTAMENTO SOBRE EXPERIÊNCIA EM DESENVOLVIMENTO DE SOFTWARE

SOBRE EXPERIÊNCIA PROFISSIONAL

1. Atua ou já atuou em projetos de desenvolvimento de software?

- Sim, por mais de 5 anos
- Sim, num período entre 2 e 5 anos
- Sim, por menos de 2 anos
- Nunca atuou em projeto de desenvolvimento de software

2. Qual foi sua principal função em projeto de desenvolvimento de software?

(mais de uma opção pode ser assinalada)

- Analista de requisitos
- Desenvolvedor de software
- Líder técnico
- Programador de software
- Projetista de software
- Web *designer*
- Outras

Caso tenha marcado a opção "Outras", favor indicar quais no espaço abaixo:

SOBRE EXPERIÊNCIA COM MODELOS DE SOFTWARE

Neste questionário estamos considerando a UML como notação de modelagem de software, independente da abordagem de desenvolvimento. Caso considere importante destacar outros modelos e notações relacionadas ao desenvolvimento de software, por favor, faça essa indicação na última seção do questionário, na resposta da pergunta 12.

3. **Em relação à sua experiência com modelos durante o desenvolvimento de software, você:** (mais de uma opção pode ser assinalada)

- Não possui experiência em CONSTRUÇÃO de modelos no desenvolvimento de software

Não possui experiência no USO de modelos no desenvolvimento de software

Possui experiência em CONSTRUÇÃO de modelos no desenvolvimento de software

Possui experiência no USO de modelos no desenvolvimento de software

Observação: O que entendemos com "USO DE MODELOS"?

Entendemos USO de modelos qualquer forma de consumo de modelos criados por outras pessoas. Ou seja, o uso de modelos pode estar relacionado a evoluir um modelo, construir outro modelo a partir daquele consumido, usar um modelo criado por outra pessoa como apoio a discussão entre diversas pessoas e outros usos identificados no processo de desenvolvimento de software.

SOBRE CONSTRUÇÃO DE MODELOS DE SOFTWARE

(Se marcou a opção "Possui experiência em CONSTRUÇÃO de modelos no desenvolvimento de software" na pergunta anterior, favor responder as próximas 2 perguntas)

3.1) Se possuir experiência em CONSTRUÇÃO de modelos no desenvolvimento de software, poderia indicar quais modelos? (Descreva sua experiência na CONSTRUÇÃO de modelos, indicando os modelos ou diagramas que já construiu)

- ☐ Modelo de atividades
- ☐ Modelo conceitual
- ☐ Modelo de casos de uso
- ☐ Modelo de dados
- ☐ Modelo de classes
- ☐ Diagrama de sequência
- ☐ Diagrama de estados
- ☐ Outros

Caso tenha marcado a opção "Outros", favor indicar quais no espaço abaixo:

3.2) Se possuir experiência em CONSTRUÇÃO de modelos no desenvolvimento de software, poderia indicar em que momento do processo de desenvolvimento os modelos foram utilizados?

SOBRE USO DE MODELOS DE SOFTWARE

(Se marcou a opção "Possui experiência em USO de modelos no desenvolvimento de software" na pergunta anterior, favor responder as próximas 2 perguntas.)

3.3) Se possuir experiência no USO de modelos no desenvolvimento de software, poderia indicar quais modelos? (Descreva sua experiência no USO de modelos, indicando os modelos ou diagramas que já utilizou)

- ☐ Modelo de atividades
- ☐ Modelo conceitual
- ☐ Modelo de casos de uso
- ☐ Modelo de dados
- ☐ Modelo de classes
- ☐ Diagrama de sequência
- ☐ Diagrama de estados
- ☐ Outros

Caso tenha marcado a opção "Outros", favor indicar quais no espaço abaixo:

3.4) Se possuir experiência em USO de modelos no desenvolvimento de software, poderia indicar em que momento do processo de desenvolvimento os modelos foram utilizados?

SOBRE EXPERIÊNCIA COM FERRAMENTAS DE MODELAGEM DE SOFTWARE

4. Quais ferramentas de modelagem de software utiliza ou utilizou?

(Indique ferramentas utilizadas com o objetivo de construir modelos de software. Mesmo que a ferramenta não esteja listada, peço indica-la no campo "Outros")

- ☐ IBM Rational Software Architect
- ☐ Enterprise Architect
- ☐ Astah (antigo JUDE)
- ☐ Visual Paradigm
- ☐ Outras

Indique aqui outras ferramentas de modelagem de software usadas no mercado que você conheça:

SOBRE EXPECTATIVAS EM RELAÇÃO AO MODELO DE SOFTWARE

5. **Qual sua intenção/expectativa com um modelo quando o está CONSTRUINDO**?(mais de uma opção pode ser assinalada)

- ☐ Que seja lido por outras pessoas
- ☐ Que seja evoluído por outras pessoas
- ☐ Que seja usado como objeto de discussão entre outras pessoas

Outras expectativas e intenções relacionadas ao modelo quando você está CONSTRUINDO o modelo (Caso tenha mais comentários, observações que complementem a resposta acima, pode ser indicado neste espaço)

6. **Durante a CONSTRUÇÃO do modelo, você considera as pessoas que vão USAR o modelo?**

- ☐ Sim
- ☐ Não
- ☐ Às vezes

Poderia comentar a resposta acima?

7. **E quando você USA modelos, qual a sua expectativa mais comum em relação a quem CONSTRÓI os modelos?**

- ☐ Que tenha menos cuidado que você quando constrói modelos
- ☐ Que tenha o mesmo cuidado que você quando constrói modelos
- ☐ Que tenha mais cuidado que você quando constrói modelos

8. **Quando você percebe "problemas de modelagem" enquanto USA um modelo, qual a sua tendência de comportamento?**

□Acha que foi um lapso pontual e segue usando o modelo normalmente

□Acha que pode haver outros problemas e segue usando o modelo, porém verificando se o mesmo está correto em outros pontos.

□Outras

Caso tenha marcado a opção "Outras", favor indicar quais no espaço abaixo:

9. **Se você "não entende" o modelo enquanto está USANDO, o que você faz?**

□Retorna a quem construiu o modelo para esclarecer dúvidas

□Segue com os seus entendimentos

□Ignora a porção "não entendida" do modelo e segue usando o modelo

□Outras

Caso tenha marcado a opção "Outras", favor indicar quais no espaço abaixo:

Poderia comentar a resposta acima?

10. **Sobre o uso de modelos de software: Em sua opinião, durante o processo de desenvolvimento, os modelos:**

Atrapalham 1□2□3□4□5□ Ajudam

Poderia comentar a resposta acima?

11. **Gostaria de relatar alguma experiência com modelos UML no contexto de projetos de desenvolvimento?**

SOBRE OUTRAS NOTAÇÕES E MODELOS

(Nesta seção você pode indicar sua experiência sobre outras notações e modelos diferentes dos abordados nas seções anteriores)

12. Sobre outras notações e outros modelos, você tem experiência com outras notações ou modelos?

☐ Sim

☐ Não

Se a resposta foi "sim", poderia indicar tais notações e modelos?

Prezado participante, você estaria disposto a participar de uma entrevista sobre esse questionário?

A entrevista seria realizada presencial ou virtualmente, usando *Skype* ou *Hangout*, de acordo com a preferência do participante. A entrevista teria uma duração aproximada de 30 minutos e seria um follow-up das suas respostas ao questionário, buscando enriquecer ainda mais os dados para a pesquisa.

☐ Sim

☐ Não

☐ Talvez

AGRADECIMENTO

Desde já agradeço a sua colaboração através das respostas ao questionário ou estendendo sua participação na entrevista. Esses dados são essenciais e de grande importância para a pesquisa em desenvolvimento.

Atenciosamente, Juliana Jansen Ferreira.

8.2
Apêndice B

ROTEIRO DE ENTREVISTA PÓS-QUESTIONÁRIO

Roteiro de Entrevista – versão 1

Foco: Levantamento sobre a experiência com a construção e uso de modelos no processo de desenvolvimento de software

Sobre relação PRODUTORES e CONSUMIDORES de modelos de software

- Quando está construindo modelos, pensa nas pessoas que vão usar o modelo?
- Coloca-se no lugar quem vai consumir os modelos que você está construindo?
- Quando constrói modelo, pensa nos usuários desse modelo?

Sobre processo de construção do modelo

- Você constrói modelos "aos poucos", evoluindo o modelo em etapas?
- Por quê? O que considera para essa evolução?
- Como sabe que um modelo está "em construção" e não finalizado? Quando considera um modelo finalizado?
- Considera que outras pessoas podem evoluir um modelo que você construiu? Em que situações?

Sobre intenção\destino de USO de modelos de software

- Considera o objetivo de USO do modelo quando o está construindo? Que tipos de uso outros podem fazer do modelo?
- E quando está USANDO, procura essa "mensagem" de quem construiu o modelo? Se o modelo tem a mesma intenção que você espera dele?

Sobre USO de modelos de software:

- Que recursos usam para "estudar" os modelos construídos por outra pessoa? (Ferramentas de modelagem, impresso, outras formas)
  - De acordo com a resposta, explorar a escolha de recursos, procurando identificar características da representação mais "confortável" ao usuário.
  - Características para LEITURA do modelo.

Roteiro de Entrevista – versão 2

O conteúdo abaixo foi adicionado gerando a versão 2 do roteiro de entrevista.

Sobre ferramentas usadas para interagir com modelos de software

- Verificar indicação de "outras" no questionário - modeladores ou desenhadores?
- Quando usa esse tipo de ferramenta e **não está muito seguro ou familiar com a notação** a ser usada para a construção de um modelo, você:

  - Acredita que a ferramenta vai te oferecer o suporte necessário para construir o modelo de acordo com a notação enquanto constrói o modelo
  - Procura outras referências da própria para te apoiar como help, conteúdo extra no site do fornecedor e etc.
  - Procura outras referências sobre o modelo e notações desejados fora da

ferramenta (Quais seriam? Pedir exemplo.)

- Pede ajuda de outras pessoas que tenham mais experiência com o modelo desejado.

- Na sua experiência, as suas expectativas em relação às ferramentas são sempre atendidas? Quando não atende suas expectativas, o que faz? (Troca de ferramenta, Adapta as suas necessidades e segue usando a mesma ferramenta, Adapta a ferramenta às suas necessidades? Como?).
- Pedir para comentar sobre experiências de uso das ferramentas - Problemas recorrentes, Soluções de contornos, Necessidade de uso de outro software para apoiar a modelagem (ex. Editor de texto para indicação de problemas de escrita, visualização completa de informações, etc.).

8.3
Apêndice C

TERMO DE CONSENTIMENTO DE ENTREVISTA PÓS-QUESTIONÁRIO

TERMO DE CONSENTIMENTO PARA APLICAÇÂO DE QUESTIONÁRIO

Título da pesquisa: Comunicação através de modelos no contexto de desenvolvimento de Software

Pesquisadora: Juliana Jansen Ferreira (Programa de Doutorado em Informática, PUC-Rio)

Caro participante, sou aluna do Programa de Pós-Graduação em Informática da PUC-Rio, tendo como orientadora a Professora Clarisse Sieckenius de Souza, e gostaria de contar com sua participação na minha pesquisa de doutorado.

Sobre a pesquisa:

Minha pesquisa tem por objetivo investigar a utilização de modelos no desenvolvimento de software como objetos de comunicação entre a equipe envolvida no processo de desenvolvimento. A contribuição esperada desta pesquisa é o aprimoramento dos processos de comunicação ao longo do ciclo de desenvolvimento de software, com o uso de modelos já construídos e utilizados durante este processo. A proposta é apresentar oportunidades de adequação de ferramentas, indicações sobre escolha e uso de notações e modelos e procedimentos de uso de tais modelos, de forma que o processo de comunicação entre os envolvidos no desenvolvimento de software tenha seus ruídos minimizados. Com menos falhas de comunicação durante o processo de desenvolvimento, várias das situações de retrabalho e as inconsistências de artefatos produzidos podem ser prevenidas e causar assim menor impacto na construção do software.

Este estudo será composto por uma entrevista de 'follow-up' com participantes que responderam o questionário para identificar o perfil do participante e sua experiência com modelos de software ao longo de sua atuação em projetos de desenvolvimento de software e concordaram em participar da entrevista. Para esta entrevista, solicito seu consentimento para a participação voluntária, tendo a duração aproximada prevista 30 minutos. Para que você decida se quer colaborar, é importante fornecer algumas informações gerais:

1) Os tipos de dados coletados são:

   i. Gravação do áudio da entrevista;
   ii. Captura de foto ou vídeo (mantendo o anonimato dos participantes, sem registro do rosto dos mesmos);
   iii. Artefatos, anotações e assemelhados, em papel ou meio eletrônico, que venham a ser produzidos durante a entrevista.

2) Toda pesquisa pauta-se nas as **normas éticas de conduta em pes**quisa, que garantem o **respeito à privacidade e ao anonimato** dos participantes que consentirem em participar. Também respeitando tais normas, os dados coletados destinam-se **estritamente** a atividades de investigação científica e desenvolvimento técnico, e são vistos e usados **exclusivamente** pela equipe de pesquisadores envolvida na pesquisa em questão. Esses dados podem ser solicitados pelo respectivo participante a qualquer momento da pesquisa. Além disto, ao divulgar os resultados do estudo em foros científicos, me comprometo rigorosamente a manter **sigilo** sobre quaisquer dados que ponham em risco o anonimato dos colaboradores e das empresas envolvidas.

3) A pesquisa não oferece nenhum risco ao participante e o consentimento para colaborar é uma escolha livre, que pode ser revisto sempre que o participante assim desejar. O participante pode desistir de colaborar com a pesquisa a qualquer momento sem que lhe traga qualquer prejuízo ou constrangimento. Eu, Juliana Jansen Ferreira, estou, portanto, disponível para prestar esclarecimentos e tirar quaisquer dúvidas sobre os procedimentos de pesquisa durante todo o período de sua execução. Para contato, fale diretamente comigo durante as atividades do projeto, ligue para o telefone 99432-5470, ou se preferir use o e-mail *jferreira@inf.puc-rio.br*. Para contatar a professora Clarisse Sieckenius de Souza, ligue para 3527-1500 ramal 4344, ou se preferir use o e-mail *clarisse@inf.puc-rio.br*.

4) Fique à vontade para imprimir este termo para futuras referências ou esclarecimentos. Caso deseje uma cópia assinada pelos pesquisadores, favor enviar um email para jferreira@inf.puc-rio.br solicitando.

De posse das informações acima, pediria então que você, por favor, se pronunciasse acerca de seu consentimento para coleta de dados durante as etapas do estudo, marcando uma das opções abaixo e depois adicionando seu nome e assinatura.

- ☐ Li e entendi as informações neste termo
- ☐ Autorizo a coleta e uso de dados para pesquisa
- ☐ Não autorizo a coleta e uso de dados para pesquisa
- ☐ Gostaria de mais informações sobre a coleta e uso de dados

Nome do participante:

Email do participante:

Independentemente de sua decisão, agradecemos a atenção que nos dispensa.

8.4
Apêndice D

CENÁRIOS DE INSPEÇÃO USADOS NA FASE DE INVESTIGAÇÃO DA INTERAÇÃO COM FERRAMENTAS DE MODELAGEM DE SOFTWARE

1) **Cenário de PRODUÇÃO DE MODELOS – Construção do Modelo de Atividade de "Requisição de Férias":**

Um engenheiro de software entra numa equipe de desenvolvimento de software para participar do desenvolvimento de um novo Sistema de Recursos Humanos (SRH). Este profissional irá participar da fase de levantamento e especificação de requisitos, interagindo com o cliente e outros membros da equipe de desenvolvimento, enquanto é definido o escopo do domínio do negócio relacionado ao software a ser desenvolvido.

Para que o novo profissional conheça o domínio do sistema e a ferramenta de modelagem, o gerente do projeto lhe passa uma tarefa de modelagem. Em reuniões prévias com o cliente, foram coletadas informações sobre o processo de "Requisição de Férias", então, a partir desses dados coletados, o profissional deve construir um modelo de atividades que pode ser usado durante as discussões com o cliente e também como fonte de informação sobre o sistema de RH para outros membros da equipe de desenvolvimento em etapas posteriores do projeto.

A partir desse modelo que será construído pelo novo profissional, o analista de requisitos, já envolvido no projeto, precisará identificar e consolidar as regras de negócio envolvidas no processo de Requisição de Férias, que devem ser tratadas pelo novo SRH. O novo profissional é informado dessa etapa posterior à construção do modelo.

O modelo de atividades deve representar a sequência de atividades de um processo de "Requisição de Férias" em uma grande empresa. O modelo deve apresentar as seguintes atividades:

1. O Empregado solicitante preenche o formulário de requisição de férias e o submete no sistema de RH
2. O SRH verifica se existe algum conflito para o mesmo período no cronograma de férias geral da área do Empregado. O cronograma de férias

geral da área apresenta o cronograma de todos os empregados de uma determinada área da empresa. O conflito de férias acontece quando duas ou mais pessoas não podem estar ausentes no mesmo período. Normalmente, essas pessoas trabalham na mesma área e atuam na mesma função, sendo substitutas uma da outra quando necessário (peers).

3. Se não existe conflito de férias, o SRH notifica o Gerente que deve revisar e aprovar a requisição de férias.
4. Se existe conflito de férias, o SRH notifica o conflito ao Empregado solicitante, que pode ajustar o período de férias para evitar o conflito ou contatar o seu Peer para negociar o período de férias.
    a. Se o Empregado solicitante quer negociar com o seu Peer, ele envia uma mensagem para este através do SRH, propondo uma alteração no período de férias do seu Peer.
    b. O Peer pode aceitar ou rejeitar a proposta feita pelo Empregado solicitante.
5. O Gerente recebe a requisição de férias e a revisa.
6. Se o Gerente concorda com a requisição de férias do Empregado solicitante sem nenhum comentário ou necessidade de ajuste, ele aprova a requisição. O SRH notifica o Responsável por Férias do RH sobre uma nova requisição de férias aprovada, que passa a ser considerada no cronograma de férias, e também notifica o Empregado solicitante sobre a aprovação da requisição de férias.
7. Se o Gerente não concorda com a requisição de férias, ele faz um comentário no formulário de requisição e o envia de volta ao Empregado solicitante para revisão da requisição.

Após a construção do modelo de atividades do processo de Requisição de Férias, o novo profissional deve disponibilizar este modelo para o analista de requisitos já envolvido no projeto para que o mesmo tome conhecimento do processo representado e que realize a atividade de tratamento das regras de negócio relacionadas ao novo SRH.

2) **Cenário de CONSUMO DE MODELOS – Tratamento das Regras de Negócio relacionadas ao Modelo de Atividade de "Requisição de Férias":**

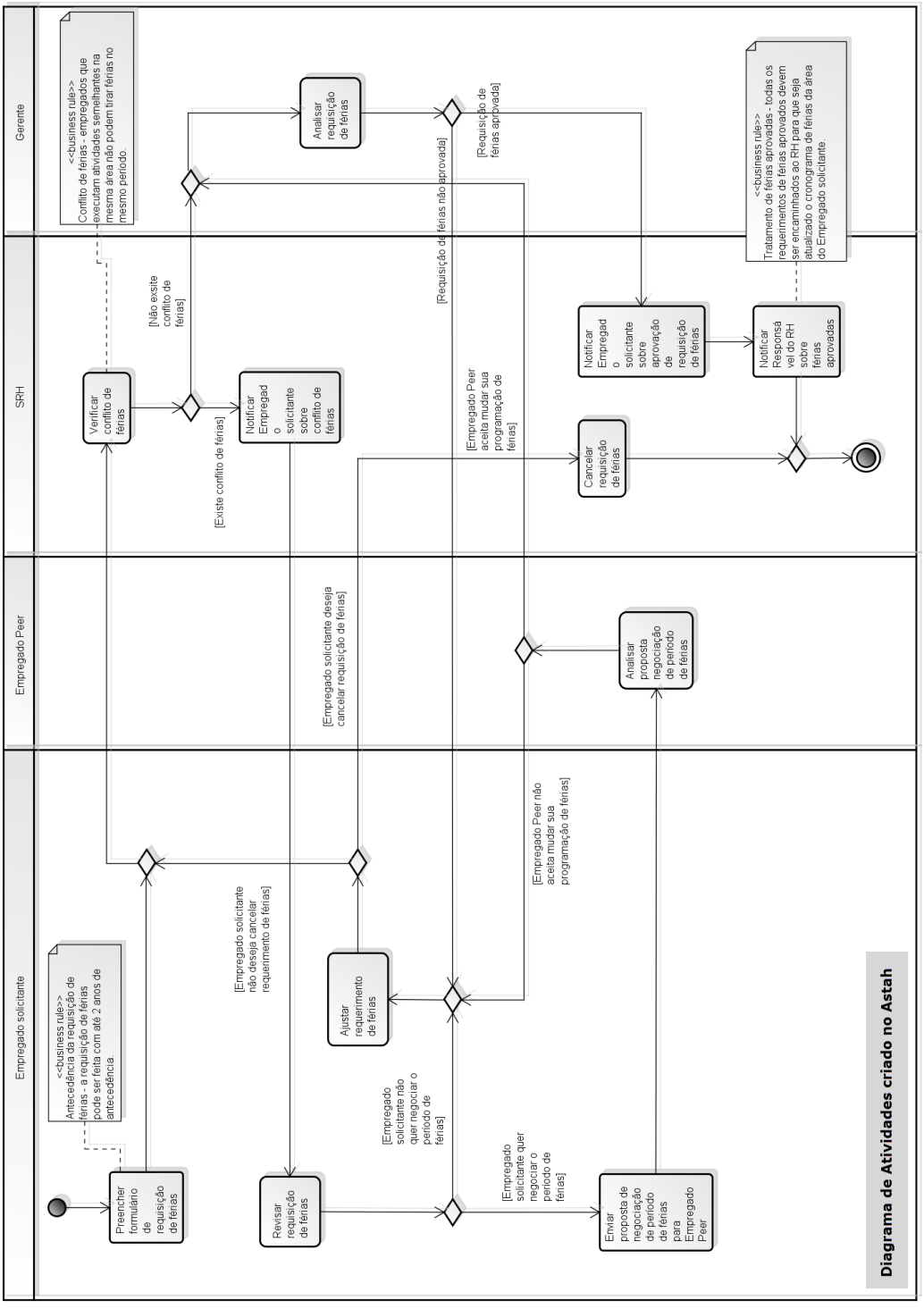
O novo profissional disponibiliza o modelo de atividades do processo de "Requisição de Férias" para o analista de requisitos, já envolvido no projeto. O analista de requisitos precisa identificar e consolidar as regras de negócio envolvidas no processo representado no modelo de atividades e que o sistema SRH deve implementar.

O analista de requisitos, então, analisa o modelo de atividades utilizando a ferramenta de modelagem de software, procura identificar as regras de negócio e as consolida para futuras consultas.

8.5
Apêndice E

EXEMPLO DE DIAGRAMA DE ATIVIDADES PRODUZIDO NA FASE DE INVESTIGAÇÃO DA INTERAÇÃO COM FERRAMENTAS DE MODELAGEM DE SOFTWARE

Diagrama de Atividades criado na ferramenta Astah



8.6
Apêndice F

TABELAS DE RELAÇÃO ENTRE CATEGORIAS E SUBCATEGORIAS DE DAS FASES DOS ESTUDOS REALIZADOS

Tabela de categorias e subcategorias da Investigação de interação com ferramentas de modelagem com número identificador associado:

Categorias de Investigação de interação com ferramentas de modelagem	Subcategorias	TNP	ID
Problemas com recursos para estender/adicionar funcionalidades	Suporte para criação de *plug-ins*	**T-P**	**1**
	Disponibilização de *plug-ins* criados por outros desenvolvedores, usuários da ferramenta	**T-P**	**2**
Problemas com recursos para construção de significado baseado em múltiplas notações	Combinação de notação UML com outras notações	**T-N-P**	**3**
	Identificação visual das regras de negócio	**T-N-P**	**4**
	Recursos de desenho no modelo	**T-P**	**5**
	Recursos para alterar representação visual de elementos	**T-N-P**	**6**
	Suporte para localização do usuário no modelo	**T-N-P**	**7**
Problemas com recursos para construção de significados baseado somente em UML	Algumas restrições da UML são implementadas	**T-N-P**	**8**
	Conhecimento sólido da UML e seus elementos	**T-N-P**	**9**
Problemas no suporte ao processo de desenvolvimento de software	Consumidor pode fazer o mesmo que o produtor	**T-P**	**10**
	Diversas opções de *menu* para exportar e importar de/para diversos formatos	**T-P**	**11**
	Mais do que um modelador, mas principalmente um modelador	**T-N**	**12**
	Modelo sempre em construção	**T-P**	**13**
	Suporte a colaboração e trabalho em equipe	**T-P**	**14**
Problemas de interação com modelos utilizando, principalmente, o *mouse*	Produção de modelos utilizando, principalmente, o *mouse*	**T-P**	**15**
	Consumo de modelos utilizando, principalmente, o *mouse*	**T-P**	**16**

Tabela de categorias e subcategorias de investigação do contexto de desenvolvimento de software associadas ao número identificador de subcategorias de investigação da interação com ferramentas de modelagem de software:

Categoria e subcategoria do Contexto de desenvolvimento de software		TNP	Relação subcategoria Investigação da Interação com ferramentas de modelagem
A	**Construção de significado através do modelo**		
A1	Conhecer UML *versus* Compreender UML	N-P	3, 4, 8, 9, 12, 14, 16
A2	Conteúdo mais importante que forma	N-P	1, 2, 9, 11, 12, 13, 14
B	**Contexto do Processo de produção e consumo do modelo**		
B1	Abandono de modelos	P	Mediada por D
B2	Confiança nos modelos	P	Mediada por D
B3	Modelos devem ser úteis	P	Mediada por D
B4	Padrão de produção de modelos	P	Mediada por D
B5	Produtores de modelos são consumidores exigentes de modelos	P	Mediada por C e D
C	**Ferramenta influenciando/impactando modelagem**		
C1	Uso de desenhadores de modelos	T-N-P	1, 2, 3, 5, 6, 9, 12, 13, 16
C2	Validação UML por ferramenta de modelagem	T-N-P	8, 9
D	**Modelagem influenciada pelo contexto de desenvolvimento**		
D1	Combinação de representações visuais e textuais	T-N-P	1, 2, 3, 4, 5, 6, 8, 11, 12, 15, 16
D2	Modelos práticos *versus* Modelos para documentar	T-N-P	1, 2, 6, 9, 11, 12, 13
D3	Nível de abstração de modelos	N-P	Mediada por D5
D4	Ponto ótimo do modelo	N-P	Mediada por D5
D5	Protocolo social para modelagem	T-N-P	1, 2, 8, 9, 11, 12, 13, 14
D6	Vida útil de modelos	P	3, 9, 12, 13
E	**Modelos como objeto de coordenação**		
E1	Objetos limítrofes do domínio como integradores	N-P	1, 2, 3, 6, 9, 11, 12, 13
E2	Produção e/ou Consumo coletivo do modelo	N-P	1, 2, 6, 9, 10, 11, 14

Categoria e subcategoria do Contexto de desenvolvimento de software		TNP	Relação subcategoria Investigação da Interação com ferramentas de modelagem
E3	Representação do todo e das partes	N-P	1, 2, 3, 7, 8, 9, 11, 12, 13