UNIVERSIDADE FEDERAL DO MARANHÃO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Adalberto Teixeira Azevedo Júnior

MobCons-AT: Uma Ferramenta de Autoria de Restrições de Mobilidade Baseada na Transformação de Modelos

São Luís
2018

Adalberto Teixeira Azevedo Júnior

MobCons-AT: Uma Ferramenta de Autoria de Restrições de Mobilidade Baseada na Transformação de Modelos

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal do Maranhão como requisito parcial para a obtenção do grau de MESTRE em Ciência da Computação.

Orientador: Luciano Reis Coutinho

Doutor em Ciência da Computação (UFMA)

Co-orientador: Francisco José da Silva e Silva

Doutor em Ciência da Computação (UFMA)

São Luís

2018

Ficha gerada por meio do SIGAA/Biblioteca com dados fornecidos pelo(a) autor(a).
Núcleo Integrado de Bibliotecas/UFMA

Azevedo Júnior, Adalberto Teixeira.
MobCons-AT: Uma Ferramenta de Autoria de Restrições de Mobilidade Baseada na Transformação de Modelos / Adalberto Teixeira Azevedo Júnior. - 2018.
119 f.

Coorientador(a): Francisco José da Silva e Silva.
Orientador(a): Luciano Reis Coutinho.
Dissertação (Mestrado) - Programa de Pós-graduação em Ciência da Computação/ccet, Universidade Federal do Maranhão, São Luis, 2018.

1. Ferrramenta de Autoria. 2. Linguagem de Modelagem de Domínio Específico. 3. Restrições de Mobilidade. 4. Transformação de Modelos. I. Coutinho, Luciano Reis. II. Silva e Silva, Francisco José da. III. Título.

Adalberto Teixeira Azevedo Júnior

MobCons-AT: Uma Ferramenta de Autoria de Restrições de Mobilidade Baseada na Transformação de Modelos

Este exemplar corresponde à redação final da dissertação devidamente corrigida e defendida por Adalberto Teixeira Azevedo Júnior e aprovada pela comissão examinadora.

Aprovada em 30 de Julho de 2018

BANCA EXAMINADORA

Luciano Reis Coutinho (orientador)

Doutor em Ciência da Computação (UFMA)

Francisco José da Silva e Silva (co-orientador)

Doutor em Ciência da Computação (UFMA)

Samyr Beliche Vale

Doutor em Informática (UFMA)

Fabio Moreira Costa

Doutor em Ciência da Computação (UFG)

Aos meus pais, irmã, professores e Cailini, sem os quais nada teria acontecido.

Resumo

Existem muitas situações em que há necessidade de monitorar em tempo real a localização e o comportamento de pessoas e/ou veículos, a fim de detectar possíveis irregularidades e controlar onde elas estão localizadas e como se movimentam, tais como nas empresas, transporte público, segurança pública, portos marítimos e aeroportos. Nesta dissertação, é apresentada a *Mobility Constraints Authoring Tool* (MobCons-AT), uma ferramenta de autoria para programação de usuário final que permite a especificação de regras de restrições de mobilidade que devem ser seguidas por dispositivos móveis. As regras são especificadas por meio de uma linguagem de domínio específico chamada *Mobility Constraints Specification Language* (MobCons-SL), parte central da ferramenta. Uma vez especificadas na MobCons-SL, essas regras são automaticamente transformadas em artefatos que permitem a detecção em tempo real das violações de restrições, através do uso de Processamento de Eventos Complexos ou *Complex Event Processing* (CEP). Essa abordagem permite a redução do tempo de entrega da solução para o cliente e o custo necessário para o desenvolvimento de soluções de software customizadas para detecção em tempo real de restrições de mobilidade. Esta dissertação também descreve o uso da MobCons-AT em cinco estudos de caso, mostrando sua aplicabilidade para diversos cenários.

Palavras-chave: ferramenta de autoria, linguagem de modelagem de domínio específico, restrições de mobilidade, tranformação de modelos.

Abstract

There are many situations in which there is a need to monitor in real-time the location and behavior of people and/or vehicles in order to detect possible irregularities and control where they are located and how they move, such as in companies, public transportation and public security. In this dissertation, Mobility Constraints Authoring Tool (MobCons-AT) is presented, an authoring tool for end-user programming that allows the specification of mobility restrictions rules that must be followed by mobile devices. The rules are specified by means of a domain specific language called Mobility Constraints Specification Language (MobCons-SL), the central part of the tool. Once specified in MobCons-SL, these rules are automatically transformed into artifacts that allow real-time detection of constraint violations by using Complex Event Processing (CEP). This approach allows the reduction in the delivery time of solution to the customer and necessary cost for the development of customized software solutions for real-time detection of mobility constraints. This dissertation also describes the use of MobCons-AT in five case studies, showing its applicability for diverse scenarios.

Keywords: authoring tool, domain-specific modeling language, mobility constraints, model transformation.

Agradecimentos

Inicialmente a Deus, pela minha vida e por todas as graças recebidas.

Ao meu orientador, o Prof. Luciano Reis Coutinho, pelo voto de confiança, apoio, compreensão e orientação. Já éramos amigos antes e, com certeza, nossa amizade sai mais fortalecida apoś esta jornada.

Ao meu co-orientador, Francisco José da Silva e Silva, pelas dicas valiosas e palavras de apoio, mesmo às vezes sendo um tanto severas (porém, sempre necessárias).

Aos professores Fábio Moreira Costa (UFG) e Samyr Beliche Vale (UFMA), por aceitarem participar da banca e avaliar este trabalho.

Ao professor Marcos Paulino Roriz Júnior (UFG), por sempre estar disposto a contibuir nos trabalhos, seja participando de reuniões on-line, opinando no projeto e criticando nos artigos.

Aos meus familiares, pelo apoio que me deram nesta trajetória. Eles me fazem querer ser alguém melhor todos os dias.

Ao nobres colegas do LSDi-UFMA, com os quais compartilhei alegrias e conhecimento. Em especial a Fernando Benedito, parceiro no projeto MobileAMP e aluno brilhante de Iniciação Científica, com grande futuro na vida, tenho certeza.

À UFMA e ao PPGCC pela estrutura dada a execução deste trabalho. Especial agradecimento ao ex-coordenador do PPGCC, Prof. Mário Meireles, ao atual, Prof. Francisco José da Silva e Silva e aos professores pela dedicação desses profissionais.

À Cailini da Silva Lima, dedicada e amada esposa, por ter mudado a minha vida quando eu mais precisava. Você foi um presente de Deus. A ideia deste mestrado foi sua, portanto, ele é seu.

"Você só sabe a força que tem quando sua última alternativa é ser forte."

Johnny Depp

Lista de Figuras

Lista de Tabelas

Lista de Siglas

API *Application Programing Interface*.

ATL *Atlas Transformation Language*.

CEP *Complex Event Processing*.

CWM *Common Warehouse Metamodel*.

DSL *Domain Specific Language*.

DSML *Domain Specific Modeling Language*.

EMF *Eclipse Modeling Framework*.

EPA *Event Processing Agent*.

EPL *Event Processing Language*.

EPN *Event Processing Network*.

FSML *Framework-Specific Modeling Languages*.

GPML *General Purpose Modeling Language*.

GPS *Global Positioning System*.

M2C *Model-to-Code*.

M2M *Model-to-Model*.

M2T *Model-to-Text*.

MBA *Model-Based Architecture*.

MBT *Model-Based Testing*.

MDA *Model-Driven Architecture*.

MDD *Model-Driven Development*.

MDE *Model-Driven Engineering*.

MDSD *Model-Driven Software Development*.

MobCons-AT *Mobility Constraints Authoring Tool*.

MobCons-SL *Mobility Constraints Specification Language*.

MOF *Meta-Object Facility*.

OCL *Object Constraint Language*.

OMG *Object Management Group*.

SMS *Short Message Service*.

SUS *System Under Study*.

SysML *Systems Modeling Language*.

UML *Unified Modeling Language*.

Sumário

1 Introdução

As cidades modernas possuem uma dinâmica cujo entendimento é uma tarefa realmente desafiadora. A análise de dados de contexto (localização, velocidade, temperatura, etc.) tem sido utilizada como uma ferramenta de grande utilidade para o entendimento da rotina da vida dos moradores dessas cidades, como se comportam e como se locomovem, sendo utilizada em vários domínios como planejamento urbano, geografia e transportes. Dada a sua importância prática, organizações governamentais têm estudado e observado tais dados de contexto em termos do uso espaço-temporal das cidades por seus cidadãos, observações estas muitas vezes realizadas por simples questionários, o que requer tempo, esforço e recursos.

O avanço das tecnologias de computação móvel e pervasiva abre oportunidade para realizar esse monitoramento em tempo real, o que melhora significativamente o tempo de resposta na análise dos dados, comparando com os métodos convencionais. Esta análise pode ser aplicada tanto no setor público quanto no setor privado para detecção de irregularidades e redistribuição de recursos.

Em muitas situações, é necessário monitorar a localização e o comportamento de pessoas e/ou objetos para detectar possíveis irregularidades e controlar onde as pessoas, grupos de pessoas ou veículos estão localizados e como eles se movem [55]. Por exemplo, uma empresa de mineração pode estar interessada em restringir o acesso de funcionários a determinadas áreas, dependendo do uso obrigatório de equipamentos de segurança, treinamento prévio ou mesmo da função exercida. Da mesma forma, essa empresa pode restringir o acesso e o comportamento de um veículo em determinada área geográfica, limitando sua velocidade, a proximidade com outros veículos ou monitorando se eles estão em rotas definidas. No transporte público, pode-se querer verificar se uma rota de ônibus está dentro de uma determinada área e seguindo um conjunto de padrões, como respeitar um limite de velocidade, rotas, atrasos, etc., o que pode ser útil para uma redistribuição de recursos, ou seja, os ônibus poderiam ser alocados com base em sua velocidade média, por exemplo. Na área de segurança pública, os veículos policiais poderiam ser monitorados para controlar sua área de cobertura e circulação, horário

de trabalho e localização. Da mesma forma, empresas portuárias podem desejar restringir acesso de funcionários a áreas restritas por motivos de segurança, além de monitorar o deslocamento dos navios. Finalmente, aeroportos podem controlar a circulação de pessoas e/ou veículos em áreas de circulação de aeronaves. Ou seja, é grande o campo de aplicações onde ter o controle do deslocamento e comportamento de dispositivos móveis é foco principal.

A detecção em tempo real de tais informações é um desafio porque requer algoritmos capazes de processar um grande volume de dados [13]. No transporte público ou no cenário de segurança pública, por exemplo, uma grande cidade com uma grande frota de veículos tem um grande fluxo de dados para processar. Atualmente, a maioria das abordagens usa o processamento em lote [44,46], em que os dados de *streaming* são armazenados em *buffer* durante um período e posteriormente processados por algoritmos *off-line* que, por sua vez, adicionam um atraso para detectar e reagir aos cenários. O Processamento de Eventos Complexos ou *Complex Event Processing* (CEP) é uma abordagem que vendo sendo utilizada para tais situações [25,26], pois é capaz de lidar com grandes fluxos de dados em alta velocidade.

Várias soluções propostas para o problema foram desenvolvidas seguindo a abordagem tradicional de engenharia de *software* [2–5, 8, 24, 26, 29, 30, 41, 43, 48]. Alternativamente, uma abordagem que está ganhando mais espaço no mercado e que dispensa a necessidade de contratar equipes de desenvolvimento, reduzindo o tempo de entrega e os custos, é o uso de ferramentas de autoria, sistemas que permitem a produção de conteúdo por parte do usuário, ou seja, os usuários se tornam autores sem a necessidade de desenvolver linhas de código.

Aliada a essa rapidez proporcionada pelas ferramentas de autoria, a Engenharia Dirigida por Modelos ou *Model-Driven Engineering* (MDE) se apresenta como um paradigma que também permite rapidez no desenvolvimento, com qualidade e baixos custos [15], baseada no uso de modelos como objeto central do processo. Com base nesse cenário e visando abstrair a complexidade do domínio ao usuário final, a MDE apresenta entre seus conceitos o de linguagem de modelagem ou *Domain Specific Modeling Language* (DSML), através da qual se torna mais simples a especificação de regras com definição gráfica e/ou textuais de alto nível, para serem convertidas em linguagem de baixo nível, de modo que não haja erros de sintaxe ou de digitação.

Este trabalho de pesquisa apresenta a *Mobility Constraints Authoring Tool* (MobCons-AT), uma ferramenta de autoria para programação pelo usuário final que faz parte do projeto MobileAMP[1] e permite a especificação de regras que definem restrições de mobilidade para dispositivos móveis, com geração automática de código e detecção de violações em tempo real, através de sua integração com a biblioteca `MobCobns`, também parte integrante do projeto citado. A MobCons-AT implementa uma DSML chamada *Mobility Constraints Specification Language* (MobCons-SL), parte central da ferramenta, que permite aos usuários modelarem graficamente e textualmente cenários que envolvam esse domínio. A MobCons-AT, juntamente com a infraestrutura desenvolvida no projeto MobileAMP, fornece uma solução geral para o problema de especificação de restrições de mobilidade a dispositivos móveis.

1.1 Objetivos

A pesquisa à qual se refere esta dissertação de mestrado tem como objetivo geral e principal desenvolver uma linguagem específica de domínio que possibilite ao usuário final, em alto nível, a especificação de restrições de mobilidade (velocidade, pontualidade, distância mínima e máxima, acesso, permanência e rota) a dispositivos móveis, grupo e tipo de dispositivos ou áreas geográficas e que permita, em tempo real, via interface com a biblioteca `MobCons`, a leitura de dados de contexto (localização, velocidade, *timestamp*, etc.) de modo a determinar se tais restrições foram violadas. Dessa forma, a linguagem proposta apoia o monitoramento, a detecção de possíveis irregularidades e o controle da movimentação e de como se comportam os dispositivos móveis.

Com o intuito de demonstrar a viabilidade da proposta, este objetivo é concretizado a partir dos seguintes objetivos específicos:

1. Fazer levantamento do estado da arte em ferramentas para monitoramento de dispositivos móveis que implementem uma linguagem de domínio;

2. Propor uma linguagem de alto nível para definição de restrições de mobilidade de dispositivos móveis;

[1]MobileAMP: `http://www.lsd.ufma.br/~mamp`

3. Implementar ferramenta de autoria para especificar restrições de mobilidade, através da utilização da linguagem proposta;

4. Desenvolver estudos de caso que permitam avaliar e aperfeiçoar a solução proposta.

1.2 Estrutura da Dissertação

Esta dissertação está organizada como segue:

- O Capítulo 2 apresenta os principais conceitos utilizados para o estudo e desenvolvimento desta pesquisa;
- O Capítulo 3 discute trabalhos relacionados, julgados mais relevantes, mostrando seus pontos fortes, fracos e mostrando a lacuna preenchida por esta pesquisa;
- O Capítulo 4 detalha a ferramenta desenvolvida, apresentando inicialmente uma visão geral do projeto MobileAMP, ao qual esta pesquisa está vinculada, descrevendo ainda as sintaxes abstrata e concreta da DSML implementada, bem como sua lógica de tranformação e geração automática de código. Finalmente, é apresentado o protótipo da ferramenta;
- O Capítulo 5 apresenta cinco estudos de caso de utilização da ferramenta proposta, mostrando como cada cenário seria modelado visualmente e abrindo discussões sobre cada caso (considerações, pontos fortes e fracos). Ao final, escolheu-se um dos cenários para que seu artefato resultante fosse submetido a testes de eficácia junto à biblioteca `MobCons`, a fim de garantir a correta instanciação das regras CEP e detecção das violações. Todos os artefatos gerados pelo processo de transformação podem ser encontrados na Seção Apêndice;
- O Capítulo 6 mostra como foi feito o processo de avaliação experimental da ferramenta proposta, assim como as análises e discussões dos resultados obtidos;
- O Capítulo 7 apresenta as conclusões obtidas nesta pesquisa e apresenta trabalhos futuros que podem ser desenvolvidos a partir deste esforço.

2 Conceitos Fundamentais

Neste capítulo são apresentados conceitos importantes para o desenvolvimento desta pesquisa. Serão discutidos tópicos relacionados a Ferramentas de Autoria, Engenharia Dirigida por Modelos, Linguagens de Modelagem de Domínio Específico, Transformações de Modelos e Processamento de Eventos Complexos.

2.1 Ferramentas de Autoria

Autoria é a faculdade de criar, gerar conteúdo. É o processo que permite a um usuário ser autor de um determinado conteúdo ou programa, dentro de um domínio específico. Sendo assim, podemos definir que linguagem de autoria é a linguagem de programação relacionada a uma ferramenta de *software* para que um usuário, leigo ou não, possa manipular, ou seja, criar, alterar ou excluir conteúdo. É através da linguagem que o *software* de autoria define as atividades apropriadas para cada nível de competência do usuário [21], ou seja, quanto mais técnico for o usuário, mais recursos a linguagem de autoria poderá disponibilizar no *software*.

Uma linguagem de autoria tem seu foco de utilização centrado no usuário de um determinado domínio que, em geral, mas não obrigatório, é um profissional leigo em programação de computadores. Embora não necessite ter um amplo domínio de computação ou programação, é exigido dele uma expertise no conteúdo a ser desenvolvido. Sendo assim, de uma boa linguagem de autoria, é esperado que seja mais intuitiva de se utilizar do que uma linguagem de programação convencional. Além disso, deve possuir uma ampla gama de recursos extras embutidos, tais como: editor gráfico, elementos multimídia (sons, imagens e vídeos), editores de texto, além de outros [21].

O objetivo maior de um sistema de autoria é o de integrar funcionalidades de forma a tornar o ambiente eficaz ao uso. A utilização de ferramentas diversas e de interface gráfica facilita o processo, mas apenas isso não é suficiente. Faz-se necessária uma boa integração entre objetos disponíveis (regras da linguagem, imagens, textos,

etc.) para se ter todas estas informações apresentadas segundo uma dinâmica relevante [47]. Encontrar o ponto de equilíbrio entre a facilidade de uso de um sistema de autoria e o potencial de aplicação de uma linguagem de autoria é uma tarefa difícil. Se o profissional tem um perfil mais técnico, existe um maior número de possibilidades a serem oferecidas, já que ele pode trabalhar com recursos mais elaborados. Mas caso esse profissional seja menos técnico, opta-se por ferramentas mais simples, voltadas ao público leigo. Contudo, tornar a linguagem de autoria menos complexa e menos potente pode levar a sistemas ineficazes. Somente um equilíbrio entre a linguagem oferecida e a interface do software utilizado para produzir os conteúdos pode levar a sistemas eficazes.

Os pesquisadores há muito se dedicam ao desenvolvimento de sistemas de autoria adaptativos e inteligentes [39]. Murray [38] classificou as ferramentas de autoria em várias categorias, como Sequenciamento e Planejamento de Currículo, Estratégias de Tutoria, Sistema Especialista de Domínio, Múltiplos Tipos de Conhecimento e Propósito Específico. Em especial, a categoria de Propósito Específico é especializada em tarefas ou domínios específicos e pode apoiar melhor as necessidades do autor em domínios específicos. Possui os seguintes pontos fortes, limites e variações:

- Pontos fortes:
  - Forte orientação de autoria;
  - Design fixo ou princípios pedagógicos podem ser aplicados.
- Limitações:
  - Cada ferramenta é limitada a um tipo específico de autor;
  - Inflexibilidade de representação e pedagogia.

As ferramentas de autoria de Propósito Específico podem suportar melhor as necessidades do autor para situações específicas. A criação é muito mais semelhante a modelos do que em outras categorias de ferramentas de autoria. Um problema potencial com a criação desse tipo de ferramenta é que, uma vez que uma tarefa e sua abordagem tenham sido codificados o suficiente para se tornar um modelo, o sistema resultante reflete uma abordagem muito particular para representar essa

tarefa, ficando assim definida para um público de autoria limitado. Murray denomina essa abordagem de um "modelo de tarefa forte para construir ferramentas de criação ricas em conhecimento" [38].

2.2 Engenharia Dirigida por Modelos

O processo de criação de *software* envolve fatores empíricos e está em constante evolução. Os avanços conquistados ao longo de mais de quatro décadas só foram possíveis, porque profissionais vêm desenvolvendo pesquisas na área de engenharia de *software*. Os engenheiros de *software*, responsáveis por tais avanços, vêm pesquisando novos métodos e ferramentas para permitir que o desenvolvimento possa ser feito com rapidez, qualidade e, acima de tudo, com baixos custos [15]. Nesse contexto, surge a abordagem baseada em modelos conhecida como MDE, que permite o gerenciamento da complexidade utilizando modelos para suportar as tarefas do desenvolvimento, com o intuito de aumentar a produtividade e preservar os investimentos [12].

Nas últimas décadas, foram propostas muitas técnicas e linguagens de modelagem para apoiar o projeto e o desenvolvimento de sistemas complexos. Muitas dessas linguagens foram definidas no contexto de abordagens metodológicas, tais como processos estruturados, orientados a objetos ou unificados, com o objetivo de facilitar e compartilhar uma visão comum e coerente do sistema em estudo e, conseqüentemente, facilitar a comunicação entre as partes interessadas. No entanto, mais recentemente surgiu uma nova tendência de abordagens considerando modelos não apenas como artefatos de documentação, mas como artefatos centrais no processo de engenharia de *software*. Além dos benefícios referidos acima, ele também permite, através de técnicas complexas como meta-modelagem, transformação de modelos, geração de código ou interpretação de modelos, a criação ou a execução automática de sistemas de software baseados nesses modelos. Essas propostas, tais como a Arquitetura Dirigida por Modelos ou *Model-Driven Architecture* (MDA), as fábricas de *software* ou recentemente a Engenharia de Linguagens de Domínios ou *DSL Engineering*, foram classificadas genericamente como MDE, mas também por nomes relacionados, como a Arquitetura Baseada em Modelos ou *Model-Based Architecture* (MBA), Desenvolvimento Orientado por Modelo ou *Model-Driven Development* (MDD),

Desenvolvimento de Software Orientado por Modelos ou *Model-Driven Software Development* (MDSD) ou Teste Baseado em Modelos ou *Model-Based Testing* (MBT). Independentemente do termo adotado e da aplicação específica, todos compartilham conceitos comuns que precisam ser abstraídos, discutidos e compreendidos.

2.2.1 Modelos

No contexto do MDE, definimos "o sistema como um conceito genérico para designar uma aplicação de *software*, plataforma de *software* ou qualquer outro artefato de *software*". Além disso, como sugerido na Figura 2.1, um sistema pode ser composto de outros subsistemas e um sistema pode ter relações com outros sistemas (por exemplo, um sistema pode se comunicar com outros) [15].

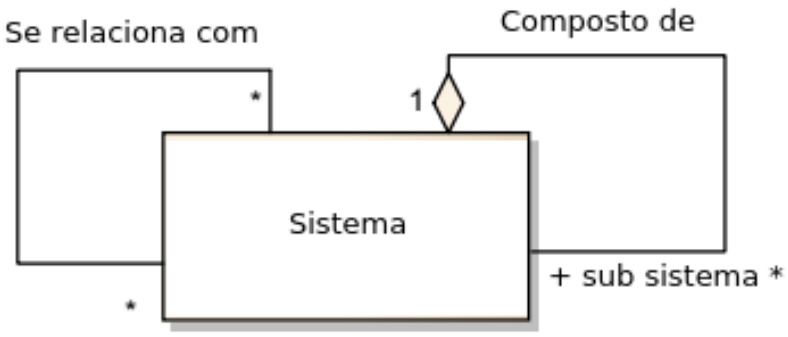

Figura 2.1: Definição de Sistema [15].

Conforme descrito na Seção 2.1, a especificação de restrições de mobilidade pode ser feita através do uso de ferramentas de autoria, uma vez que se trata de um domínio específico. Para fazer isso, os problemas de domínio são descritos por um modelo. Em geral, um modelo representa uma visão parcial e simplificada de um sistema. Portanto, a criação de múltiplos modelos é geralmente necessária para representar e entender melhor o sistema em estudo. Os modelos permitem compartilhar uma visão e conhecimento comuns entre os atores técnicos e não técnicos, facilitando e promovendo a comunicação entre eles. Além disso, os modelos tornam o planejamento do projeto mais eficaz e eficiente, proporcionando uma visão mais adequada do sistema a ser desenvolvido e permitindo que o controle do projeto seja alcançado de acordo com critérios objetivos.

Um modelo é uma abstração de um Sistema em Estudo ou *System Under Study* (SUS), também conhecido como "Universo do Discurso"ou apenas "sistema", que pode já existir ou se destina a existir no futuro [15]. No processo de desenvolvimento de um sistema, quais os elementos-chaves do problema? Como eles estão relacionados? Quais limitações eles apresentam? Qual arquitetura deve ser usada? Por meio da utilização de modelos, essas questões são compreendidas antes do desenvolvimento. Dessa forma, pode-se afirmar que modelos são elementos essenciais para lidar com a complexidade da realidade, visto que o produto é originado com base nos modelos criados [11].

A importância de modelos se tornou tão relevante que abordagens como MDE procuram declarar que tudo em seu contexto são modelos [10]. Por exemplo, modelos são utilizados para descrever os artefatos de um *software* [11]. Assim, pode-se considerar que os modelos passaram de atores coadjuvantes para atores principais dentro desse âmbito de desenvolvimento [9,15].

Alguns conceitos de modelos apresentados na literatura: Rodrigues [15] define modelo como: “um sistema que ajuda a definir e a dar respostas do sistema em estudo, sem a necessidade de considerá-lo diretamente”; enquanto Mellor et al. [36] definem modelo, como: “um conjunto de elementos que descrevem alguma realidade física, abstrata ou hipotética”; já Cabot et al. [10] conceitua modelos como “uma representação simplificada ou parcial da realidade, definida para realizar uma tarefa ou chegar a um acordo”.

A partir dessas definições, há um consenso de que um modelo define um SUS e vice-versa. No entanto, um modelo é em si um sistema, com sua própria identidade, complexidade, elementos, relações, etc. Em particular, quando pensamos em um modelo de modelo, temos que considerar que um deles desempenha o papel do sistema em estudo e, conseqüentemente, é ele mesmo um sistema. Para resumir, e como sugerido na Figura 2.2, definimos "modelo como um sistema que ajuda a definir e dar respostas do sistema em estudo sem necessidade de considerá-lo diretamente" [15].

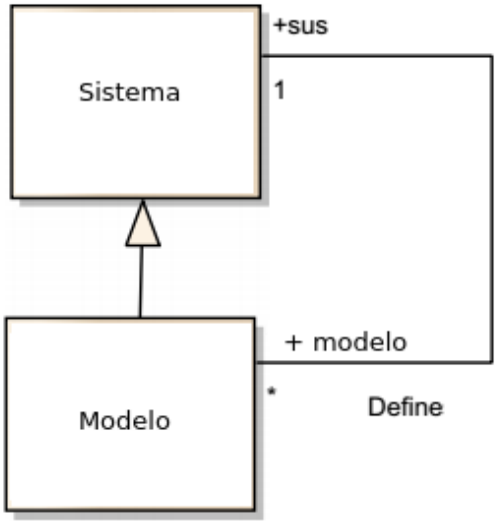

Figura 2.2: Definição de Modelo [15]

2.2.2 Metamodelos

Os modelos são os principais artefatos em abordagens dirigidas por modelos. Assim é natural que eles sejam definidos conforme outros modelos. Para a *Object Management Group* (OMG)[2], modelos que descrevem outros modelos são chamados de metamodelos e a relação entre um modelo e seu metamodelo pode ser definida como uma relação de instanciação, ou seja, um modelo é uma instância de seu metamodelo. A Figura 2.3 apresenta a relação entre modelo e metamodelo.

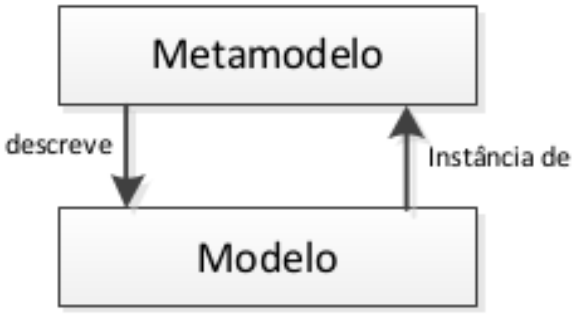

Figura 2.3: Relação Modelo x Metamodelo [15].

Metamodelos são essenciais em abordagens dirigidas por modelos, porque eles especificam a forma que os modelos podem ser construídos. Dessa forma, Mellor et al. [36] conceituam metamodelo como "um modelo de uma linguagem de

[2]OMG: http://www.omg.org/index.htm

modelagem que define a estrutura, a semântica e as restrições para uma família de modelos". Cabot et al. [10] afirma que "um metamodelo constitui basicamente a definição de uma linguagem de modelagem, uma vez que fornece uma maneira de descrever toda a classe de modelos que podem ser representados por essa linguagem".

Um problema bem conhecido e recorrente de metamodelagem é como definir o metamodelo inicial. Se um metamodelo é um modelo de linguagem de modelagem, deve haver um meta-metamodelo que descreva sua linguagem de modelagem, e assim por diante, em níveis superiores e meta-metamodelos mais abstratos. A solução comum para superar esse problema é usar uma linguagem que, em um determinado nível dessa hierarquia, se descreva em sua própria linguagem. Existem vários exemplos desta solução. No campo das linguagens naturais, a língua inglesa se descreve em inglês ao nível da definição de gramática, dicionários, etc. No campo das linguagens de programação, Lisp é um exemplo bem conhecido de uma linguagem que se descreve, proporcionando, em particular, um compilador Lisp escrito em Lisp.

No campo das linguagens de modelagem, a solução proposta pela OMG, baseada em uma arquitetura de quatro camadas e diretamente suportada através da *Meta-Object Facility* (MOF), é um exemplo popular. No topo dessa hierarquia há a camada de meta-metamodelagem (designada como M3) que é principalmente responsável por fornecer uma linguagem para especificar metamodelos. MOF é uma camada de meta-metamodelo única porque é instanciada a partir de seu próprio modelo, ou seja, o MOF é definido no MOF. Na camada abaixo (designada como M2), os metamodelos são definidos por instanciação do meta-metamodelo (isto é, cada elemento do metamodelo é uma instância de um elemento definido no meta-metamodelo). A *Unified Modeling Language* (UML) e a *Common Warehouse Metamodel* (CWM) são alguns exemplos desses metamodelos, ou seja, exemplos de instâncias MOF. Na camada abaixo da M2 (designada como M1), os modelos são definidos de acordo com o interesse e as necessidades de seus usuários: tipicamente para diferentes domínios de aplicação e diferentes níveis de abstração (p.ex. ao nível da definição de negócios, requisitos técnicos ou design de software). Finalmente, o nível mais baixo da hierarquia (a camada M0) contém instâncias reais de elementos definidos no modelo que realmente existem no contexto de um ambiente computacional ou mesmo no mundo real (por exemplo, um vídeo específico em seu próprio *laptop*). A

Figura 2.4 apresenta a arquitetura de quatro camadas da OMG aplicada para modelos construídos usando MDE.

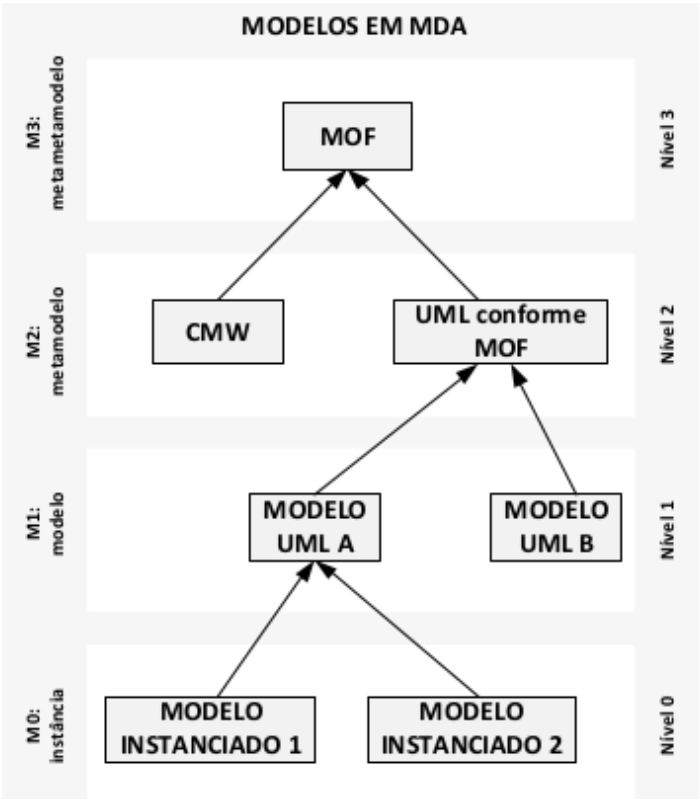

Figura 2.4: Arquitetura de quatro camadas da OMG (com adaptações).

Uma forma mais simples de compreender essa estrutura em camadas é considerar que independente do nível da arquitetura ao qual se esteja referindo, um modelo deve estar conforme a seu metamodelo. Diz-se que um modelo está conforme seu metamodelo quando todos os elementos descritos no metamodelo podem ser instanciados pelo modelo [10]. Dessa forma, M0 é uma instância de M1, M1 é uma instância de M2, M2 é uma instância de M3 e M3 é uma instância dele mesmo, pois seria possível haver infinitas dessas relações, mas é senso comum na literatura não passar desse nível de abstração.

2.3 Linguagens de Modelagem de Domínio Específico

A simplicidade com que o usuário define suas especificações pode ser melhor obtida através de uma DSML, também conhecida como *Domain Specific Language* (DSL), ou seja, uma linguagem de programação de computadores de

expressividade limitada focada em um domínio específico. Para Harel et al, DSL é um conjunto de todos os modelos possíveis, definidos por um metamodelo, que estejam de acordo com uma sintaxe abstrata, representada por uma ou mais sintaxes concretas e que satisfaçam uma determinada semântica [20], conforme demonstrado na Figura 2.5.

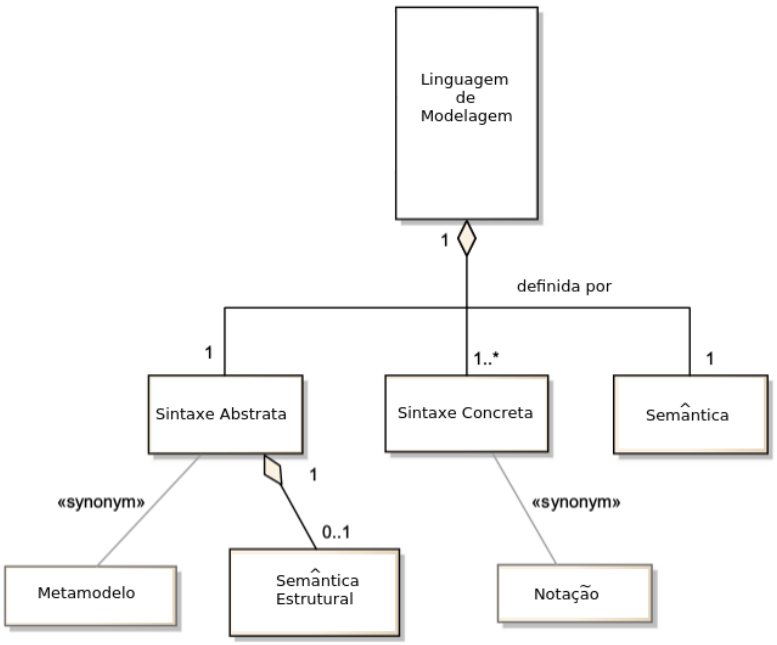

Figura 2.5: Definição de DSML (com adaptações) [15].

2.3.1 Sintaxe Abstrata

A definição de uma linguagem de modelagem geralmente começa capturando e identificando o domínio principal da aplicação (conceitos, abstrações e relações). Esta tarefa representa a fase de análise de domínio, para construir a linguagem de modelagem [37]. Trata-se principalmente de um exercício de abstração e conceituação e, em seguida, síntese do conhecimento do domínio, que o arquiteto da linguagem de modelagem deve saber ou deve obter através da interação direta com os especialistas do domínio. O resultado dessa atividade produz a sintaxe abstrata da linguagem de modelagem, que corresponde a um metamodelo com todos os conceitos identificados no nível do domínio.

A sintaxe abstrata define todos os nomes dos conceitos identificados e suas respectivas relações, por isso é importante que esses nomes estejam próximos do domínio da aplicação para serem facilmente entendidos por seus usuários. Por outro

lado, sua simplicidade deve ser promovida, por exemplo, para evitar a introdução de conceitos adicionais com termos incompreendidos que embora possam torná-la mais expressiva, podem também torná-la mais difícil de usar e manter.

Conforme ilustrado na Figura 2.5, a sintaxe abstrata ainda inclui a semântica estrutural (estática) que se concentra principalmente na definição de regras de ligação entre seus elementos. Por exemplo, e de forma simplificada, a semântica estrutural permite definir que um elemento do tipo A pode estar relacionado a outros elementos do tipo B de acordo com essa ou aquela restrição; Ou pode-se determinar que os elementos do tipo A nunca podem estar relacionados a elementos do tipo C. Em geral, a semântica estrutural de uma linguagem de modelagem pode ser descrita de diferentes maneiras: seja através de uma linguagem de restrições declarativas (por exemplo, *Object Constraint Language* (OCL)[3] no caso da UML, através da especificação informal de linguagem natural, ou através de uma abordagem de mistura. Por exemplo, a semântica estrutural da própria UML é descrita extensivamente em linguagem natural, diagramas de classes e alguns aspectos definidos em OCL.

2.3.2 Sintaxe Concreta

A sintaxe concreta de uma linguagem de modelagem refere-se à sua notação, ou seja, como os usuários aprenderão e usarão, seja lendo ou escrevendo e projetando os modelos. Assim, o sucesso de uma linguagem de modelagem dependerá do equilíbrio certo entre simplicidade, expressividade, capacidade de escrever, legibilidade, aprendizado e eficácia [53].

Podem ser fornecidas diferentes notações, sejam gráficas, textuais, tabulares e baseadas em formulários, ou combinações delas. Em geral, as notações gráficas tendem a ser mais adequadas para ilustrar as relações entre conceitos, mudar os valores em uma distribuição espacial ou temporal, sequências causais e temporais entre eventos ou fluxos de dados e controles em cenários de modelagem de processos. No entanto, os modelos gráficos não são tão escaláveis quanto os modelos de texto ou tabulares, o que significa que eles não são os mais adequados para suportar grandes modelos. Também são pouco aplicáveis para escrever ou visualizar expressões lógicas ou ações complexas - para tal, as notações textuais tendem a ser mais apropriadas [53].

[3]OCL: http://https://www.omg.org/spec/OCL/

Finalmente, uma vez que uma linguagem de modelagem pode ter múltiplas sintaxes concretas, seria possível combinar adotar uma notação textual para escrita/autoria e uma notação gráfica apenas para leitura/visualização, o que facilitaria a usabilidade da linguagem.

2.3.3 Semântica

Em geral, a semântica revela o significado de expressões (ou modelos) sintaticamente válidas especificadas em uma determinada linguagem. Para linguagens naturais, isso significa correlacionar sentenças e frases com conceitos, pensamentos e sentimentos com base em nossas experiências e antecedentes. Para linguagens de programação, a semântica descreve o comportamento que um computador deve seguir ao executar um programa nessa linguagem. Esta especificação pode descrever a relação entre entrada e saída de um programa ou pode fornecer uma explicação passo-a-passo sobre como um programa será executado em uma máquina virtual ou real [15].

Para linguagens de modelagem, e de acordo com os conceitos e modelos envolvidos, existem dois tipos de semântica: semântica executável e não executável [22]. A semântica executável diz respeito a conceitos diretamente relacionados a linguagens de programação, ou seja, relacionados à ordem de execução de programas, como os encontrados em máquinas de estado, seqüência e diagramas de atividades. Por outro lado, a semântica não executável diz respeito a conceitos que não estão diretamente relacionados com a execução do *software* (ou qualquer outro tipo de execução), como os conceitos envolvidos na implantação de componentes de software em nós de hardware (por exemplo, componentes UML e diagramas de implantação), ou especificação dos requisitos do usuário (por exemplo, através de diagramas de casos de uso UML).

2.3.4 Classificação

As linguagens de modelagem podem ser classificadas em dois tipos: além da já citada DSML, existem as Linguagens de Modelagem de Propósito Geral ou *General Purpose Modeling Language* (GPML) [18, 35, 37, 51], cada uma com suas características e finalidades distintas.

Uma GPML é caracterizada por ter um maior número de construções genéricas, o que incentiva um uso mais amplo e generalizado em diferentes campos de aplicação. UML e *Systems Modeling Language* (SysML)[4] são exemplos populares desse tipo de linguagem, fornecendo grandes conjuntos de construções e anotações usadas para especificar e documentar, respectivamente, sistemas de *software* de acordo com o paradigma orientado a objetos, ou qualquer tipo de sistemas, conforme entendido pela disciplina de engenharia do sistema.

Uma DSML, por sua vez, tende a usar algumas construções ou conceitos mais próximos do seu domínio de aplicação. Uma vez que ela é expressa usando conceitos de domínio, normalmente é mais fácil ler, entender, validar e comunicar, facilitando a cooperação entre desenvolvedores e especialistas em domínio. Além disso, alguns estudos argumentam que este tipo de linguagem pode melhorar a produtividade, confiabilidade, manutenção e portabilidade [23, 51]. Por outro lado, seu uso pode aumentar alguns problemas, como o custo de aprender, implementar e manter uma nova linguagem, bem como as ferramentas de suporte para desenvolvimento [37].

2.4 Transformações entre Modelos

Modelos podem ser criados manualmente ou gerados automaticamente por meio de um processo de conversão de um modelo de origem em um modelo de destino chamado transformação entre modelos [27]. Dessa forma, a transformação entre modelos é outro elemento essencial dentro das abordagens dirigidas por modelos, pois por meio dela é possível automatizar um ciclo de desenvolvimento de *software*, baseado em modelos [15].

As transformações entre os modelos são fundamentais, pois a partir dessas transformações podemos maximizar o reuso dos artefatos e, com isso, aumentar a qualidade e a rapidez na produção de sistemas. Esse tipo de transformação pode ser classificado em automático, semi-automático e manual. São consideradas automáticas quando não há a necessidade da intervenção humana durante o processo de geração de código; semi-automáticas quando o desenvolvedor escolhe quais elementos do modelo

[4]SysML: http://www.omgsysml.org/

serão transformados e, por fim, manuais quando é o desenvolvedor que realiza a transformação. Em um processo de desenvolvimento de software utilizando MDE é possível a definição de regras de transformação para que este processo ocorra de forma automática [6].

Uma transformação é composta por diversas definições de transformação, que Parreiras [42] define como "um conjunto de regras de transformação, que juntas descrevem como um modelo na linguagem de origem, pode ser transformado em um modelo na linguagem de destino". Para ele, uma regra de transformação é definida como "um ou mais elementos da linguagem de origem que podem ser transformados em um ou mais elementos da linguagem de destino".

Uma das formas de realizar transformações de forma automática é através da utilização de uma DSML [6]. Na literatura, existem diversas linguagens formalmente definidas para lidar com transformações de modelos, sendo uma das mais reconhecidas a *Atlas Transformation Language* (ATL), criada pela comunidade Eclipse e introduzida dentro do framework *Eclipse Modeling Framework* (EMF)[5] com o objetivo de prover as definições de transformações entre modelos dentro daquele ambiente [1].

A Figura 2.6 fornece uma visão geral da transformação ATL (*Author2Person*) que permite gerar um modelo *Person*, em conformidade com o metamodelo *MMPerson*, a partir de um modelo *Author* que esteja em conformidade com o metamodelo *MMAuthor*. A transformação projetada, que é expressa por meio da linguagem ATL, está de acordo com o metamodelo ATL. Neste exemplo, os três metamodelos (*MMAuthor*, *MMPerson* e ATL) são expressos usando a semântica do metametamodelo Ecore, que é o metamodelo central do EMF.

[5]EMF: http://projects.eclipse.org/projects/modeling.emf

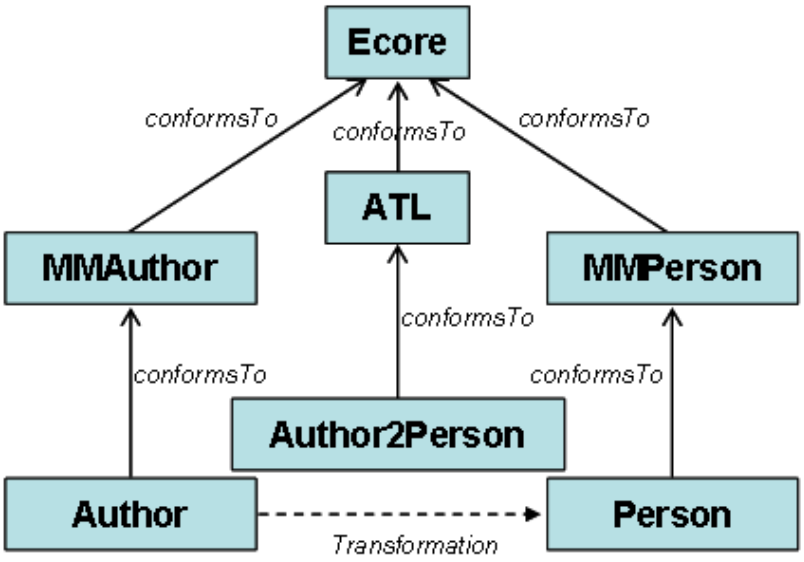

Figura 2.6: Linguagem ATL [1].

As transformações realizadas pela MDE podem, de acordo com seu resultado, serem classificadas em: Modelo-para-Modelo ou *Model-to-Model* (M2M); Modelo-para-Código ou *Model-to-Code* (M2C); Modelo-para-Texto ou *Model-to-Text* (M2T). Na primeira o artefato-alvo gerado é outro modelo, já na segunda é um código-fonte e na terceira os resultados são, além do código-fonte, outros artefatos textuais, tais como, casos de teste e diversas documentações [6, 14]).

No que tange aos tipos M2C e M2T, a técnica mais comum para produção de artefatos é a geração de código, cujo objetivo é escrever um *software* a partir de modelos abstratos e refiná-lo em código [6]. Em sua essência, um gerador de código é um utilitário que, a partir de uma especificação de alto nível de um problema implementável, transforma automaticamente essa especificação em artefatos-alvo para esse problema. Tais artefatos-alvo podem ser de vários tipos, como códigos-fonte, testes e diversos tipos de documentação [6, 14, 19].

Geradores de código podem ser baseados em três tipos de abordagens: *Framework-Specific Modeling Languages* (FSML), padrões de código e *templates* [28]. A primeira é composta por uma sintaxe abstrata e um mapeamento entre essa sintaxe abstrata e a *Application Programing Interface* (API) de um *framework-base*, identificando os conceitos do *framework-base* e os representando em conceitos de linguagem por meio da sintaxe abstrata. Por sua vez, os padrões de código utilizam uma técnica de interpretação lógica onde é possível determinar como e quando interagir com

elementos de interesse [6]. Por fim, a abordagem baseada em *templates* é a técnica mais difundida e também é uma maneira prática de gerar código-fonte utilizando modelos de alto nível. Essa técnica contém regras que são mapeadas nos artefatos-fonte, como a ferramenta Acceleo[6].

Template é um arquivo de texto padronizado qualquer, instrumentado principalmente com construções de seleção e expansão de código, sendo responsável por realizar consultas de parâmetros em uma entrada (um programa ou um arquivo textual), diagramas ou modelos. As informações obtidas por meio destas consultas são utilizadas como parâmetro para produzir código-fonte, conforme exemplificado na Figura 2.7 [6, 14]. Os *templates* são lidos pelo gerador de código (passo 1) e os parâmetros necessários são consultados nas entradas (passo 2). Por fim, essas informações contidas nos *templates* e nas entradas são processadas pelo gerador de código, resultando no código-fonte da aplicação (passo 3).

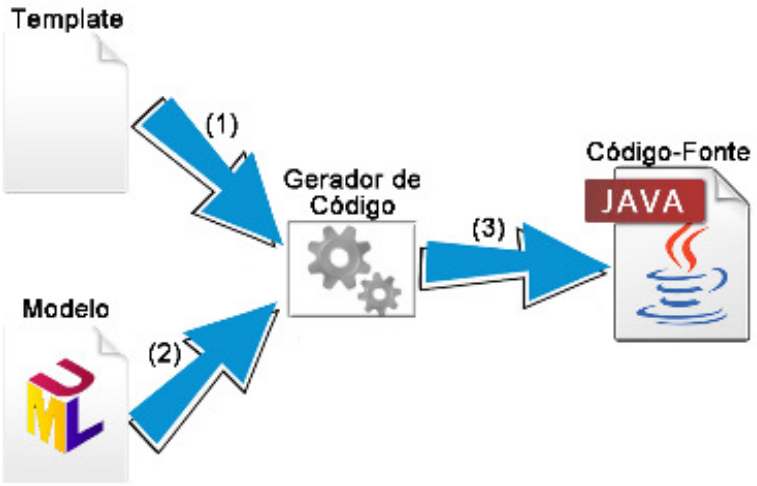

Figura 2.7: Geração de código baseada em templates [6].

2.5 Processamento de Eventos Complexos

A velocidade com que os dados estão sendo gerados tem aumentado continuamente. Em contrapartida, o tempo da resposta do serviço ao cliente tem reduzido. Soma-se a isso o fato de existirem informação em diferentes formatos, o que torna tal gerenciamento muito difícil. O grande desafio é estabelecer a integração dessa grande quantidade de dados em ambientes complexos. Neste cenário, CEP é um

[6]Acceleo: https://www.eclipse.org/acceleo/

paradigma de programação baseado em regras que agrega informações em tempo real, revelando padrões e tendências através de eventos aparentemente desconexos [17,34]. Ao contrário dos sistemas gerenciadores de banco de dados (SGBDs) tradicionais, nos quais os dados são primeiro armazenados e depois consultados, o CEP armazena consultas contínuas e extrai informações através delas, ou seja, ao invés de armazenar os dados, o CEP se concentra em analisar e processar continuamente os dados, através de consultas [25,52], conforme mostrado na Figura 2.8.

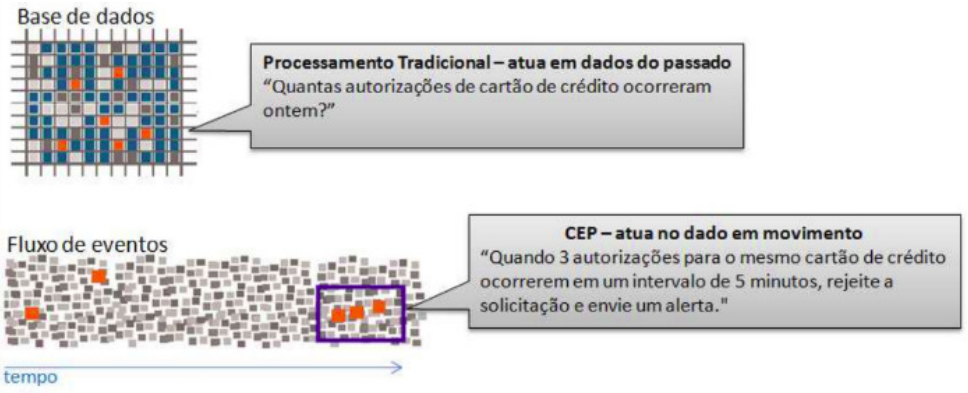

Figura 2.8: CEP x SGBDs [52]

O processamento de eventos é qualquer forma de computação que executa operações em eventos. Um evento simples (ou atômico) é algo que acontece ou é rotulado como acontecendo [16]. Já os eventos complexos podem ocorrer quando vários outros eventos simples (ou atômicos) acontecem, ou seja, um evento complexo pode ser entendido como uma estrutura composta de vários eventos atômicos. Eventos atômicos são registros imutáveis de uma ocorrência passada de uma ação ou mudança de estado. Ou ainda, pode ser uma parte (ou "pedaço") de uma informação que represente algo que aconteceu no mundo real ou em um sistema de *software*, por exemplo, mudança de valores de ações no mercado financeiro ou mudança de temperatura de certo local. Eventos estão relacionados entre si através de condições de tempo, causalidade e agregações. E dada uma sequência ordenada dos eventos através do tempo, tem-se então, um fluxo de eventos ou *Event Stream* [16].

Os eventos são criados por produtores (*producers*), que são entidades (por exemplo, sensores e aplicações clientes) que geram ocorrências de interesse de um domínio da aplicação. O fluxo de dados de eventos é o resultado de uma sequência de eventos criada e enviada por produtores. O CEP processa continuamente essa

sequência de eventos de entrada, analisa e manipula tais eventos. Em seguida, têm-se eventos derivados na saída que são entregues para entidades consumidoras (*consumers*) (por exemplo, aplicações de monitoramento). Essas saídas geralmente representam notificações sobre situação detectadas. Mais precisamente, é possível identificar hierarquias de eventos, onde cada consulta contínua é executada por um estágio intermediário de processamento CEP conhecido como *Event Processing Agent* (EPA). A partir da comunicação entre EPAs, produtores e consumidores através da interconexão entre seus terminais de entrada e saída tem-se, finalmente, uma rede de processamento de eventos ou *Event Processing Network* (EPN) [25], conforme ilustrado na Figura 2.9.

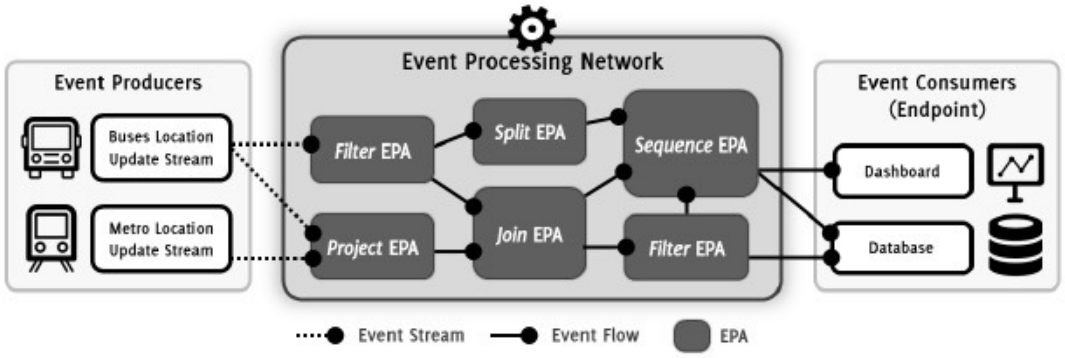

Figura 2.9: EPAs, produtores e consumidores formam uma EPN [16].

De forma geral, o processamento realizado por um *software* CEP consiste da procura por padrões de eventos, ou o desvio deles. A descrição desses padrões é passada ao CEP pelos usuários, através de uma Linguagem de Processamento de Eventos ou *Event Processing Language* (EPL), que descreve um evento complexo utilizando relações de casualidade e temporalidade aliadas aos operadores lógicos. Em CEP, são providos um rico conjunto de conceitos e operadores para processar eventos, que incluem regras, transformação de eventos, agregações, filtros e padrões, produção de eventos derivados, dentre outros. Os eventos são continuamente analisados e processados, depois manipulados. As saídas são entregues aos consumidores de eventos (aplicações de monitoramento), que são geralmente notificações sobre situações detectadas.

2.6 Síntese

Neste capítulo foram vistos conceitos importantes para o desenvolvimento desta pesquisa, tais como Ferramentas de Autoria, Engenharia Dirigida por Modelos, Linguagens de Modelagem de Domínio Específico, Transformações de Modelos e Processamento de Eventos Complexos.

Inicialmente foi mostrado que as ferramentas de autoria de Propósito Específico podem suportar melhor as necessidades do usuário e do autor para geração de conteúdo em situações específicas. Em outras palavras, essa categoria de ferramentas de autoria pode ser baseada em domínios específicos e, assim, modelada por linguagens.

Portanto, o problema de especificar restrições de mobilidade pode ser tratado por ferramentas de autoria. Para isso, esse problema é descrito por um modelo, que é uma abstração de um sistema frequentemente usada para substituir o sistema em estudo, permitindo a compreensão de várias questões abordadas entes do processo de desenvolvimento. Recentemente, os modelos se tornaram mais do que artefatos de documentação, assumindo funções centrais no processo de engenharia de *software*, conhecido como MDE.

A simplicidade com que o usuário especifica as restrições pode ser mais facilmente obtida através de uma linguagem de modelagem ou DSML, que é uma linguagem de expressividade limitada focada em um domínio específico, construída de acordo com uma sintaxe abstrata, representada por uma ou mais sintaxes concretas e que satisfaçam uma determinada semântica. As DSMLs usam algumas construções ou conceitos mais próximos de seu domínio de aplicação, o que geralmente as tornam mais fáceis de ler, entender, validar e comunicar.

Uma vez elaborada a especificação de restrições de mobilidade por uma DSML, ela pode gerar artefatos por meio de um processo de transformação. Os modelos podem ser criados manualmente ou gerados automaticamente por meio de um processo de conversão de um modelo de origem em um modelo de destino chamado transformação entre modelos. A M2T é uma das principais transformações que geram ou produzem artefatos de software, geralmente código-fonte e outros arquivos de texto, a partir de modelos.

Ao final do processo de transformação, são gerados artefatos que instanciam regras responsáveis pela conversão das restrições de mobilidade definidas pelo usuário em consultas CEP. Trata-se de uma solução baseada em regras que agrega informações em tempo real, revelando padrões e tendências por meio de eventos aparentemente desconectados, analisando e processando continuamente os dados e cujas saídas são entregues às aplicações de monitoramento através de notificações sobre as situações detectadas.

3 Trabalhos Relacionados

A análise dos trabalhos relacionados nesta dissertação foi realizada, primeiramente, por meio de um estudo sobre os projetos anteriores que fazem parte do mesmo projeto de pesquisa. Em seguida, foi feita uma revisão sistemática (metodologia específica para pesquisa) para coletar e avaliar as evidências disponíveis relacionadas ao tema em estudo. A revisão sistemática consiste na análise dos trabalhos relacionados a este que estão indexados em bases de dados de grande impacto na área de Computação e na análise das dissertações e teses.

No contexto desta pesquisa, não foi encontrado nenhum relato na literatura de uma ferramenta de autoria que implemente uma DSML para especificar restrições de mobilidade em dispositivos móveis. Nesta seção, são apresentados trabalhos relacionados que permitem o monitoramento de dispositivos móveis, sem implementar uma linguagem específica para o domínio.

Há alguns trabalhos que investigam como adquirir a posição geográfica de um veículo e enviá-la para um nó central que exibe o local ou o envia via *Short Message Service* (SMS) [2, 5, 8]. Outros trabalhos também podem detectar e alertar quando uma unidade monitorada excede um certo limite de velocidade [3, 24] ou quando eles deixam uma determinada área [43], ou mesmo em ambos os casos [4]. Também foi encontrado um trabalho que emite alertas quando ocorre um desvio de rota [41]. Em [26], é apresentado um protótipo de sistema que monitora padrões de localização e mobilidade de um grande número de objetos móveis em grandes cidades. Esse sistema adotou o CEP para busca otimizada em bases de dados. Lee et Ruy [29] apresentaram um sistema de gerenciamento veicular capaz de gerenciar e obter localizações de veículos em ambientes móveis, via *Global Positioning System* (GPS), em tempo próximo ao real. Também foram encontrados trabalhos para busca de anomalias em movimentos de embarcações, tais como em [48], onde os dados são analisados via GPS, e em [30], onde é apresentado um *framework* para modelagem que facilita a aprendizagem do comportamento dos movimentos através de características dos dados extraídos das trajetórias.

A maior parte destes trabalhos está focada no rastreamento de veículos, com exceção do [43] que foi desenvolvido para rastrear pessoas, [41] que foi desenvolvido para rastreamento de cargas e [30, 48], que rastreim embarcaçõs. Deve-se também observar que alguns associam restrições de mobilidade (limites de velocidade, área limitada ou rota) às unidades monitoradas, além de [24] associar um limite de velocidade a uma área específica (ou estrada).

A Tabela 3.1 abaixo mostra um comparativo entre os trabalhos relacionados a esta pesquisa.

Tabela 3.1: Tabela Comparativa de Trabalhos Relacionados

	Tempo Real?	Emite Alerta?	Especifica Restrições?	Localização?	DSML?
1. Hybrid gps-gsm localization of automobile tracking system [1]	X			X	
2. An event-based bus monitoring system [4]	X			X	
3. Design and development of a low cost ubiquitous tracking system [5]	X	X		X	
4. Remote monitoring of vehicle diagnostics and location using a smart box with global positioning system and general packet radio service [2]	X	X	X	X	
5. Ubiquitous gps vehicle tracking and management system[3]	X	X	X	X	
6. A novel security enabled speed monitoring system for two wheelers using wireless technology [20]		X	X	X	
7. Swtrack: An intelligent model for cargo tracking based on off-the-shelf mobile devices [32]	X	X	X	X	
8. Protection of the child/elderly/ disabled/pet by smart and intelligent gsm and gps based automatic tracking and alert system [34]	X	X	X	X	
9. Mobiiscape: Middleware support for scalable mobility pattern monitoring of moving objects in a large-scale city [22]	X	X		X	
10. Design of vehicle information management system for effective retrieving of vehicle location [23]	X	X		X	
11. A framework for anomaly detection in maritime trajectory behavior [24]	X	X	X	X	
12. Anomaly detection in maritime data based on geometrical analysis of trajectories [39]	X	X	X	X	

4 A Ferramenta de Autoria MobCons-AT

Dada a ausência de um trabalho que suporte a especificação de restrições de mobilidade de dispositivos móveis através de uma ferramenta de autoria baseada em uma DSML, ausência esta evidenciada no Capítulo 3, buscou-se neste estudo preencher tal lacuna. Portanto, neste capítulo é apresentada a MobCons-AT, uma ferramenta de autoria para programação pelo usuário final que faz parte do projeto MobileAMP e permite a especificação de regras que definem restrições de mobilidade para dispositivos móveis, com geração automática de código e detecção de violações em tempo real via CEP. Através da implementação de uma DSML chamada MobCons-SL, parte central da ferramenta, a MobCons-AT permite aos usuários modelarem graficamente e textualmente cenários que envolvam esse domínio.

O capítulo inicia dando uma visão geral do projeto MobileAMP, uma vez que este estudo é sua parte integrante. Posteriormente, são detalhados a DSML desenvolvida e o processo de tranformação para geração de artefatos. Por fim, é apresentado um protótipo para usuário final.

4.1 O Projeto MobileAMP

O projeto MobileAMP tem seu foco no desenvolvimento de *software* que permite a análise em tempo real de padrões de mobilidade e de atividades a partir do processamento de fluxos de dados de contexto gerados a partir de pessoas, veículos e outras entidades móveis (também conhecidos como dispositivos ou nós móveis). Ao contrário da maioria dos trabalhos anteriores, que usam uma abordagem estatística para armazenar dados e processá-los *off-line*, o MobileAMP se destaca ao descobrir esses padrões em tempo real, incorporando primitivas de processamento de fluxo de dados CEP. A arquitetura geral proposta pelo projeto é composta por 4 componentes principais, conforme mostrado na Figura 4.1.

Atualmente, o projeto apresenta uma biblioteca chamada `MobCons` que pode manipular fluxos de dados de posição de entrada gerados por entidades móveis e

aplicar regras de restrições de mobilidade como primitivas do CEP para monitorá-las. Este estudo faz uso da biblioteca `MobCons`, através da ferramenta de autoria MobCons-AT, cujo objetivo é permitir a especificação de restrições de mobilidade em alto nível, pela implementação da linguagem MobCons-SL. As especificações criam regras que são transformadas em artefatos responsáveis por acessar as classes implementadas na biblioteca `MobCons` para, enfim, instanciar as regras CEP.

Figura 4.1: Projeto MobileAMP: Arquitetura Geral

- Módulo de Autoria: ferramenta de autoria que permite especificar restrições de mobilidade em alto nível, através da implementação de uma DSML. As regras criadas são transformadas em artefatos que acessam as classes `MobCons` responsáveis por instanciar as regras CEP;
- Dispositivos Móveis: representam entidades (veículos, pessoas, etc.) executando um aplicativo móvel que pode obter dados do sensor, tais como localização, velocidade, registro de data e hora e tipo do dispositivo, enviando a eles o fluxo de dados correspondente para análise posterior;

- Nuvem: módulo que possui a implementação do processador CEP. Recebe os eventos em tempo real das unidades móveis e os transmite junto com as regras violadas para o componente de exibição;

- Módulo de Monitoramento: módulo que permite monitorar dispositivos móveis e violações das restrições que foram detectadas. Violações são exibidas na forma de alertas.

4.2 Linguagem MobCons-SL

Para abstrair a complexidade de especificar restrições de mobilidade de dispositivos móveis, a ferramenta MobCons-AT implementa uma DSML chamada MobCons-SL, responsável por permitir aos usuários a modelagem de forma gráfica e textual dos cenários que envolvem esse domínio. Por esse motivo, a linguagem implementada é a parte central e mais importante da ferramenta.

A DSML criada é composta por uma sintaxe abstrata, representada por um metamodelo, e por uma sintaxe concreta implementada de acordo com esse metamodelo, representada por uma notação gráfica e complementada por uma notação textual.

4.2.1 Syntaxe Abstrata

O metamodelo proposto usa o padrão EMF em sua definição. EMF é um padrão comum para modelos de dados, no qual muitas tecnologias e estruturas são baseadas. Ele é capaz de produzir artefatos para várias linguagens, possui várias ferramentas incorporadas e também é compatível com outras ferramentas que usam esse padrão. O metamodelo é apresentado na Figura 4.2, sendo responsável por descrever a modelagem de especificação de restrições de mobilidade de dispositivos móveis.

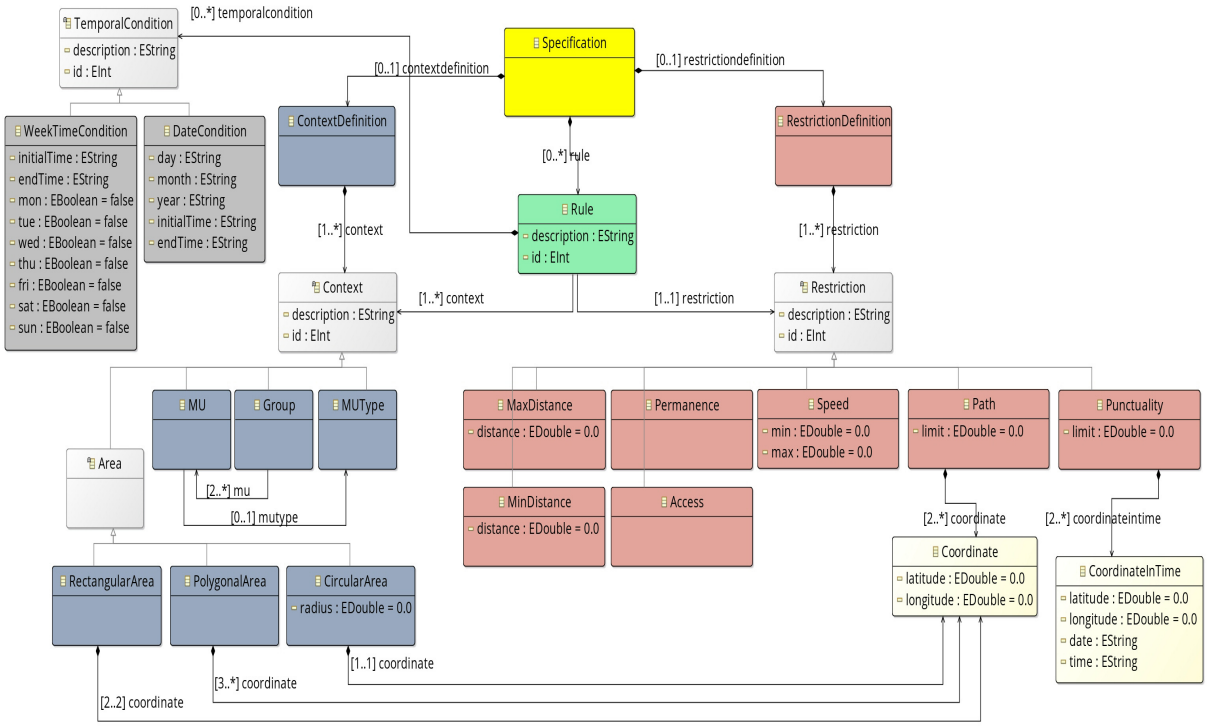

Figura 4.2: Linguagem MobCons-SL - Syntaxe Abstrata

O conjunto de classes presentes no metamodelo, separadas como contextos, restrições e condições temporais, foi definido após a análise e estudo de como modelar restrições de mobilidade em cenários como empresas mineradoras, segurança pública, portos marítimos, aeroportos e transporte público. Analisando as rotinas comumente encontradas nesses cenários e outras possíveis situações, pensou-se que seria interessante restringir as velocidades máxima e mínima dos dispositivos móveis (veículos, pessoas, navios, aeronaves, etc.), suas rotas, pontualidade, distâncias máximas e mínimas entre eles, assim como o acesso e permanência desses dispositivos em determinados locais. Uma vez definidas as restrições, o foco dos contextos foi estendido para além de dispositivos móveis isolados, abrangendo também grupos e tipos específicos de dispositivos, assim como as áreas geográficas em que eles se encontram. Finalmente, percebeu-se a necessidade de definir janelas de tempo, ou seja, condições temporais para as quais as regras podem ser feitas, permitindo desta forma obter uma maior precisão na especificação e no monitoramento.

No metamodelo proposto, a classe `Specification` é o primeiro nível e ponto de partida para a especificação de restrições. A partir da classe `Specification` é possível definir as classes `ContextDefinition`, que define

os contextos, `RestrictionDefinition`, que define as restrições de mobilidade, `TemporalCondition`, que define as condições temporais e `Rule`, que define as regras. Todas as classes presentes no metamodelo serão detalhadas nas seções seguintes.

Contextos

A classe `Context` representa os contextos, ou seja, para quais unidades móveis uma restrição deve ser aplicada, podendo ser dos seguintes tipos, de acordo com o metamodelo:

- `MU` (*Mobile Unit*): Usada para aplicar uma restrição a uma única unidade móvel, que pode ser uma pessoa, veículo, embarcação, etc.;
- `Group`: Usada para agrupar duas ou mais `MU`, de qualquer tipo (pessoa, veículo, etc.), as quais receberão a mesma restrição. Requer que todas as `MU` que a compõem sejam definidas;
- `MUType` (*Mobile Unit Type*): Define um contexto para todas as unidades móveis de um mesmo tipo específico (carros, ônibus, pessoas, etc.). Não requer a definição de `MU` e é bastante útil para modelagem de cenários de forma mais simples com um grande número de dispositivos. A informação do tipo de dispositivo é repassada através do fluxo de eventos e definida em cada aplicativo móvel;
- `Area`: Define uma área geográfica onde os dispositivos móveis estejam inseridos, que pode ser circular (`CircularArea`), retangular (`RectangularArea`) ou poligonal (`PolygonalArea`). Cada tipo de área possui suas particularidades:
  - `CircularArea`: deve ser composta por uma coordenada geográfica, que indicará seu centro, e um raio;
  - `RectangularArea`: deve ser composta por duas coordenadas geográficas, que indicarão as extremidades para cálculo da área retangular;
  - `PolygonalArea`: deve ser composta por no mínimo três coordenadas geográficas.

Restrições

A classe `Restriction` representa as restrições, ou seja, limitações impostas à mobilidade de dispositivos móveis. O metamodelo proposto permite a especificação dos seguintes tipos:

- `Access`: Restringe a mobilidade de dispositivos móveis detectando quando acessam uma área restrita, ou seja, delimita uma área na qual dispositivos móveis especificados não podem entrar. Requer a existência de dois contextos, sendo um contexto `Area` e o outro `MU`, `Group` ou `MUType`;

- `Permanence`: Restringe a mobilidade de dispositivos móveis detectando quando elas saem de uma área delimitada, ou seja, delimita uma área da qual dispositivos móveis especificados não podem sair. Requer a existência de dois contextos, sendo um contexto `Area` e o outro `MU`, `Group` ou `MUType`;

- `Speed`: Impõe um limite máximo e/ou mínimo de velocidade (em quilômetros por hora) para os dispositivos móveis especificados. Pode ser associada a qualquer contexto, e quando associada a um contexto `Area`, passa a valer a qualquer dispositivo móvel que nele se encontre;

- `Path`: Define uma rota a ser seguida pelos dispositivos móveis especificados, detectando quando eles se desviam do caminho acima de um limite (em quilômetros). Deve ser composta de pelos menos duas coordenadas geográficas e requer que o contexto associado seja `MU`, `Group` ou `MuType`;

- `MaxDistance`: Define a distância máxima (em quilômetros) permitida entre dois contextos especificados, sendo que ambos não podem ser `Area`, dado que assim a distância seria fixa. Quando especificada para contextos `Group` ou `MUType`, será considerada a distância máxima entre qualquer dispositivo móvel desses contextos para o outro contexto associado;

- `MinDistance`: Define a distância mínima (em quilômetros) permitida entre dois contextos especificados, sendo que ambos não podem ser `Area`, dado que a distância seria fixa. Quando especificada para contextos `Group` ou `MUType`, será considerada a distância mínima entre qualquer dispositivo móvel desses contextos para o outro contexto associado;

- `Punctuality`: Define o tempo exato ("dd/mm/yyyy hh:mm") em que os dispositivos móveis especificados devem passar por um conjunto determinado de pontos geográficos dentro de uma rota. Deve ser composta de pelos menos duas coordenadas geográficas e requer a existência de dois contextos, sendo um contexto `Area` e o outro `MU`, `Group` ou `MUType`;

Condições Temporais

A classe `TemporalCondition` é opcional e permite determinar janelas de tempo, ou seja, quando uma restrição de mobilidade será válida. Permite definir uma data específica (dia, mês e ano), dias da semana e/ou intervalos de tempo (horas de início de fim) para as restrições. Por exemplo, é possível definir que um ônibus deve viajar a uma velocidade máxima de 60 km/h, de segunda a sexta-feira (das 8h às 18h), 80 km/h aos sábados e domingos e 40 km/h em feriados. Um `TemporalCondition` pode ser de dois tipos:

- `WeekTimeCodition`: Define janelas de tempo nos dias da semana (de segunda a domingo), com possibilidade de informar horas de início e fim. Por exemplo, permite especificar restrições para segunda, terça, sexta e domingo, entre 09h e 17h;

- `DateCondition`: Define janelas de tempo em dias, meses ou anos específicos, com possibilidade de informar horas de início e fim. Por exemplo, permite especificar restrições para o dia 25 de dezembro de 2018, ou para todo dia 01 (independente de mês e ano), ou para todo mês 12 (independente do ano), ou ainda para todo o ano de 2018, entre 20h e 23h.

Regras

A classe `Rule` define associações entre as classes `Context`, `Restriction` e `TemporalCondition`, razão pela qual é um dos objetos mais importante do modelo, permitindo que uma classe `Restriction` seja relacionada a um ou mais `Context` e a um ou mais `TemporalCondition`, opcionalmente. É pela classe `Rule` que posteriormente será instanciada a regra CEP responsável pelo monitoramento

desejado, conforme visto na Seção 4.3. Em outras palavras, a classe `Rule` é o objeto através do qual se pode, por exemplo, definir um cenário como:

Exemplo 1: *Todos os veículos e motos (`MU`) só podem transitar dentro da Universidade Federal (`Area`) a uma velocidade máxima de 30 km/h (`Speed`), às segundas, quartas e sextas, das 6 às 11 horas (`WeekTimeCondition`).*

Para fins demonstrativos, o cenário acima será utilizado como exemplo de uso nas subseções seguintes, mais precisamente no tocante ao processo de transformação (4.3) e na apresentação do protótipo da ferramenta (4.4).

4.2.2 Sintaxe Concreta

Para ilustrar as relações entre todos os conceitos envolvidos no metamodelo e permitir de forma mais clara a alteração de valores (tais como valor da velocidade, das distâncias mínima e máxima, data/hora para condição de tempo, etc.) foi desenvolvida uma notação gráfica na qual cada objeto do metamodelo é representado por um ícone diferente. Cabe aos usuários aprender conceitos do domínio, cada representação gráfica e projetar os modelos. Buscou-se dessa forma alcançar um equilíbrio entre simplicidade, expressividade, capacidade de escrever, legibilidade, aprendizado e eficácia.

Para derivar a sintaxe abstrata em sua forma concreta é necessário reconhecer quais partes da forma concreta representam que elementos na forma abstrata. A figura 4.3 mostra a notação gráfica da lnguagem MobCons-SL, onde pode-se observar que todos os objetos presentes no metamodelo (sintaxe abstrata) estão presentes e relacionados conforme esse metamodelo. A área ilustrada na cor azul é um *container* que representa o objeto `ContextDefinition`, ou seja, é a área onde todos os contextos devem ser criados. A área ilustrada na cor vermelha, por sua vez, é um *container* que representa o objeto `RestrictionDefinition`, sendo a área onde todas as restrições devem ser criadas. Já na área externa a esses *containers* (parte central da figura), encontram-se os objetos `Rule` e `TemporalCondition`.

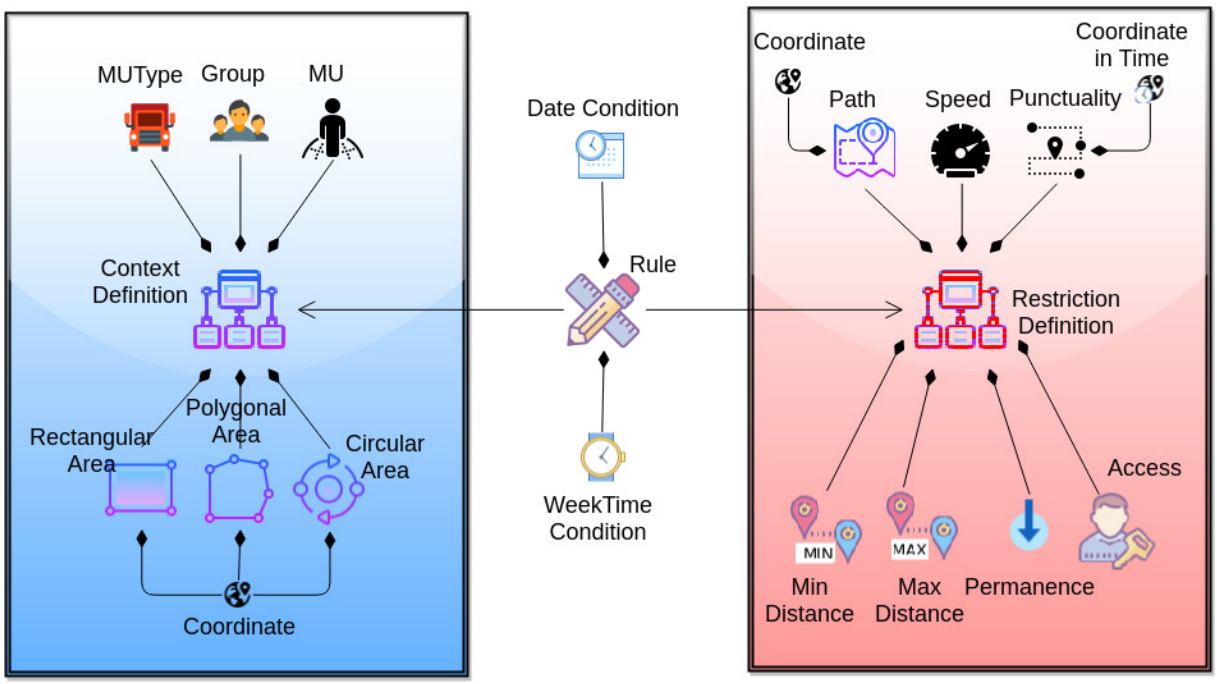

Figura 4.3: Linguagem MobCons-SL - Sintaxe Concreta

Para complementar a notação gráfica apresentada, uma notação textual se fez necessária a fim de que os atributos dos objetos modelados fossem adicionados/editados manualmente. A seguir, alguns dos valores editados de forma textual:

- `id`: identificador do objeto;
- `description`: descrição do objeto;
- `radius`: raio da área circular;
- `latitude` e `longitude`: valor das coordenadas (latitude e longitude);
- `date` e `time`: data e hora definidos para coordenadas no tempo;
- `min` e `max`: valores mínimo e máximo da velocidade permitida;
- `limit`: limite de tolerância definido para indicar permanência de uma rota;
- `distance`: valor da distância a ser mantida;
- `initialTime` e `endTime`: horas inicial e final de uma condição temporal;
- `mon`, `tue`, `wed`, `thu`, `fri`, `sat` e `sun`: dias da semana (de segunda a domingo) de uma condição temporal;

- `day`, `month`, `year`: dia, mês e ano de uma condição temporal.

4.3 Transformação

Um aspecto fundamental definido pela MobCons-AT é a possibilidade de gerar código automaticamente através do uso de transformações M2T diretamente associadas ao modelo. As regras de transformação criadas na MobCons-AT têm como entrada o modelo concreto (gerado de acordo com o metamodelo e composto de todas as especificações feitas pelo usuário através da linguagem MobCons-SL) e geram como artefatos de saída código em Java[7], cujo objetivo é instanciar as classes da biblioteca `MobCons` responsáveis por disparar as consultas CEP.

O processo automatizado de transformação da modelagem, feito a partir de um determinado modelo, para o código que será utilizado em sua implementação ocorre através de um conjunto de regras definidas. Essas regras são estruturadas na forma de *templates* e *queries*, sendo cada *template* responsável por um estágio de transformação e cada *query* usada para consultar valores ou coleções de valores definidas no modelo. As transformações usadas na MobCons-AT foram implementadas usando a ferramenta Acceleo, escolhida pelo seu fácil uso e sua integração com o Eclipse IDE através de um *plug-in*.

A transformação é iniciada pela instanciação da classe `Specification`. A partir desta classe, todas as classes `Rule` especificadas são pesquisadas e, para cada uma, o processo de transformação procura pelas classes `Restriction`, `Context` e possíveis `TemporalCondition`, conforme pode ser visto nos trechos em destaque da Figura 4.4. Em outras palavras, a rotina desenvolvida procura por todas as regras especificadas, obtendo de cada uma a restrição, contextos e condições temporais (se existirem) associados.

[7]Java: https://www.oracle.com/java/index.html

```
[for (itRule : Rule | anSpecification.rule)]
        /*
        *
        * Rule Description: [itRule.description/]
        * Rule ID: [itRule.id/]
        *
        */
    [comment

        the DISTANCE RESTRICTION has a different logic
        in MaxDistance, the param "true" indicates that the distance is MAX
        in MinDistance, the param "false" indicates that the distance is MIN

    /]
    [if (getRestrictionName(itRule.restriction).equalsIgnoreCase('MaxDistance'))]
        [distanceRestriction(itRule,getContextOrigFromDistanceRestriction(itRule),itRule.restriction.oclAsType(MaxDistance).distance.toString(),true)/]
        [contexts(itRule,getContextDestFromDistanceRestriction(itRule))/]
    [elseif (getRestrictionName(itRule.restriction).equalsIgnoreCase('MinDistance'))]
        [distanceRestriction(itRule,getContextOrigFromDistanceRestriction(itRule),itRule.restriction.oclAsType(MinDistance).distance.toString(),false)/]
        [contexts(itRule,getContextDestFromDistanceRestriction(itRule))/]
    [else]
        [comment

            the others RESTRICTIONs follow the same logic

        /]
        [restriction(itRule)/]
       [for (itContext : Context | itRule._context)]
        [contexts(itRule,itContext)/]
       [/for]
    [/if]

[/for]
```

Figura 4.4: Trecho de Código do Processo de Transformação

Para demonstrar o processo de transformação desenvolvido, o cenário de exemplo mostrado na seção 4.2.1 será usado, conforme mostrado novamente:

Exemplo 1: *Todos os veículos e motos (`MU`) só podem transitar dentro da Universidade Federal (`Area`) a uma velocidade máxima de 30 km/h (`Speed`), às segundas, quartas e sextas, das 6 às 11 horas (`WeekTimeCondition`).*

Para modelagem desse exemplo, a linguagem MobCons-SL gera os seguintes objetos: uma classe `RestrictionDefinition` e uma restrição `Speed`, uma classe `ContextDefinition` e um contexto `PolygonalArea`, composto de no mínimo três objetos `Coordinate`. A associação entre o contexto e a restrição é feita através da criação de uma classe `Rule`, composta de uma `WeekTimeCondition`. A notação gráfica que representa o modelo concreto de entrada gerado pode ser vista na Figura 4.5.

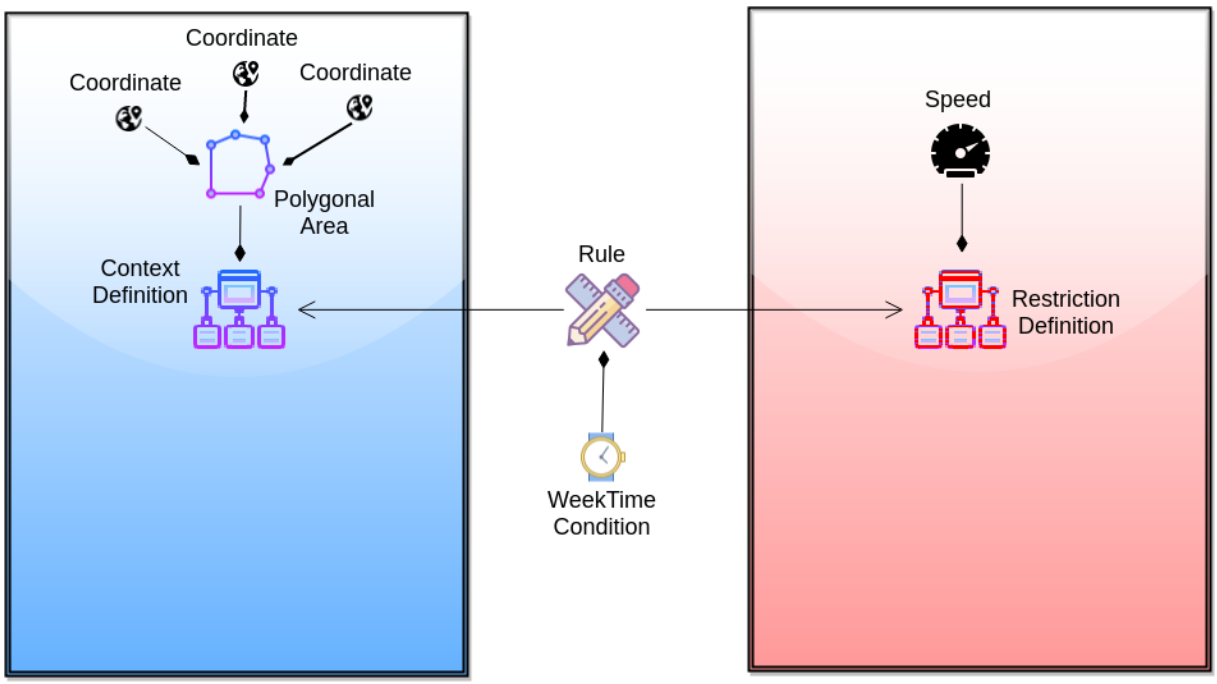

Figura 4.5: Modelo concreto de entrada, baseado no exemplo da Seção 4.2.1.

A partir deste ponto, os objetos definidos no modelo concreto de entrada são transformados nos objetos implementados na `MobCons`. O código abaixo mostra o artefato gerado após transformação do modelo concreto, onde é possível observar o mapeamento entre as classes do metamodelo e as classes da biblioteca `MobCons`, a instanciação do construtor principal da biblioteca (linhas 3 e 4) e posteriormente as chamadas aos construtores das classes `SpeedRestriction` (linha 9), `AreaContext` (linha 10) e `ConstraintTimeClause` (linha 12). A regra que liga uma restrição a um ou mais contextos e a uma ou mais condições temporais, traduzindo os objetos especificados em regras CEP, é dada pela chamada do método *newMC()* (linha 15), que é o método implementado na biblioteca `MobCons` responsável por instanciar uma nova restrição de mobilidade, buscando a(s) regra(s) CEP associada(s) à restrição e instanciando essa(s) regra(s) na *engine* CEP (algumas restrições podem ser obtidas diretamente por uma regra, tais como a velocidade, enquanto outras mais complexas requerem mais de uma regra CEP para funcionarem corretamente, tais como rota e pontualidade).

```
1  public class MobCons {
2    public static void main(String[] args) {
3      MampCoreMobconsCore mobconsCore = new MobconsCore();
4      mobconsCore.init(true);
5          /*
6          * Rule Description: Regra Exemplo 1
7          * Rule ID: 1
8          */
9      SpeedRestriction restriction11 = new SpeedRestriction(30.0);
10     AreaContext contextArea11 = new AreaContext(
11         new Point2D.Double(10.0, 30.0), 10.0);
12     ConstraintTimeClause weekTimeCondition11 =
13         ConstraintTimeClause.timeIntervalAndDaysOfWeekInArray(
14         "06:00", "11:00",2,4,6);
15     mobconsCore.newMC(restriction11, contextArea11, weekTimeCondition11);
16   }
17 }
```

Listing 1: Artefato gerado após transformação, baseado no exemplo da Seção 4.2.1.

Algumas decisões de projeto utilizadas no processo de transformação devem ser citadas, dadas as suas importâncias:

- Os arquivos de geração dos artefatos possuem extensão ".mtl", que são os arquivos principais do processo e contêm os *templates* e *queries* usados para obter informações dos modelos. No caso, foram utilzados dois arquivos nomeados "main.mtl"e "generate.mtl";
- Para evitar erros do artefato no compilador Java, os nomes das variáveis criadas durante o processo são formados da seguinte maneira: pela concatenação de um prefixo com o código identificador (id) da classe em uso, e o código identificador (id) da classe `Rule`;

  1. um prefixo, tipo texto (exemplo: restriction, contextArea, etc);
  2. código identificador (id) da classe `Restriction`, `Context` ou `TemporalCondition` em uso;
  3. código identificador (id) da classe `Rule`.

Exemplo: as variáveis **restriction11**, **contextArea11** e **weekTimeCondition11** usadas no código acima, linhas 12, 13 e 15 respectivamente. Caso houvesse uma outra restrição `Speed` (id=2), associada ao mesmo contexto de área já existente (id=1), obviamente através de uma outra `Rule` (id=2), as novas variáveis criadas seriam: **restriction22**, **contextArea12**.

- Na implementação, criou-se *templates* para geração dos *imports* e definições da classe Java, assim como para chamada das *queries* responsáveis pela consulta dos valores e coleções definidos no modelo;

- Fez-se uso de um recurso muito útil no Acceleo, chamado *Java Service*, que é um código Java, criado em uma classe pertencente ao projeto (nomeada de Utility.java), acessado diretamente pelos *templates*, ao qual é delegada a execução de instruções, neste caso, a execução de uma função para verificar quais os dias da semana especificados pelo usuário final, quando usada uma `WeekTimeCondition`. Esse tipo de recurso é útil para manipulações um pouco mais complexas dos objetos.

Como dito anteriormente, o processo de transformação é baseado em um modelo, que por sua vez é baseado em um metamodelo, gerando com isso, artefatos. Na biblioteca `MobCons`, as classes não seguem a mesma nomenclatura do metamodelo proposto, uma vez que, enquanto aquela é voltada ao desenvolvedor, este precisa de conceitos de maior clareza e fácil entendimento ao usuário final, conceitos estes que se refletirão na linguagem MobCons-SL e, posteriormente, na interafce da MobCons-AT. A Tabela 4.1 mostra o mapeamento das classes definidas no metamodelo para suas respectivas classes presentes na biblioteca `MobCons`.

Tabela 4.1: Mapeamento Metamodelo → MobCons

	Metamodelo	MobCons
Contextos	MU	SingleMUContext
	Group	GroupMUContext
	MUType	KindOfMUContext
	CircularArea	AreaContext
	RectangularArea	AreaContext
	PolygonalArea	AreaContext
Restrições	Speed	SpeedRestriction
	Path	PathRestriction
	Access	AreaRestriction
	Permanence	AreaRestriction
	MaxDistance	DistanceRestriction
	MinDistance	DistanceRestriction
	Punctuality	PunctualityRestriction
Condições Temporais	WeekTimeCondition	ConstraintTimeClause
	DateCondition	ConstraintTimeClause
Regras	Rule	newMC

Pode ser observado que buscou-se simplicidade e clareza nos conceitos presentes no metamodelo. As classes que representam contextos de área, nomeadas no metamodelo de `CircularArea`, `RetangularArea` e `PolygonalArea`, são nomeadas como `AreaContext` na `MobCons`, diferenciando-se em sua assinatura, o que poderia não ficar claro ao usuário final. Da mesma forma, as restrições `Acess` e `Permanence` presentes no metamodelo, são nomeadas como `AreaRestriction` na `MobCons`, cuja diferença também se da em sua assinatura. Por fim, as classes referentes às condições temporais foram nomeadas `WeekTimeCondition` e `DateTimeCondition` no metamodelo, enquanto na `MobCons` existe unicamente a classe `ConstraintTimeClause`.

4.4 MobCons-AT - Protótipo

A implementação da interface gráfica usada na ferramenta MobCons-AT foi feita com a ferramenta Sirius[8], escolhida pelo seu fácil uso e por sua integração com o Eclipse IDE através de um *plug-in*. Os usuários não precisam aprender conceitos externos ao seu domínio de negócios, bastando somente aprender as visualizações fornecidas pela ferramenta e como navegar entre elas.

A interface gráfica proposta permite representar visualmente os conceitos definidos na linguagem MobCons-SL. A partir da classe `Specification`, é possível modelar todos as classes `Context`, `Restriction` e `TemporalCondition` definidas para o cenário desejado, associando-as através da classe `Rule`.

Para demonstrar a interface desenvolvida, o cenário (Exemplo 1) mostrado na seção 4.2.1 será usado. A Figura 4.6 mostra o cenário modelado graficamente no protótipo.

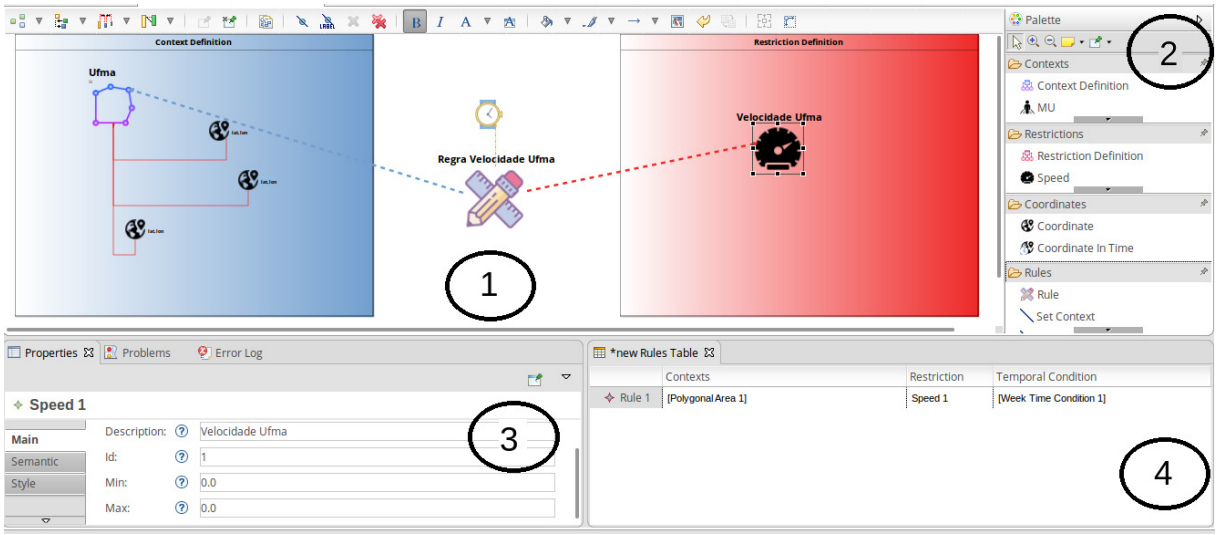

Figura 4.6: Protótipo da Ferramenta de Autoria - Mobcons-AT

Para modelar não só este, como qualquer cenário, inicialmente cria-se os objetos `ContextDefinition` e `RestrictionDefinition`, uma vez que eles servem como *containers* para os objetos `Context` e `Restriction`, respectivamente (conforme metamdelo). Para uma melhor visualização desses objetos na ferramenta, optou-se por representá-los como dois retângulos: um azul (associado à classe `ContextDefinition`) e outro vermelho (asociado à classe

[8]Sirius: https://www.eclipse.org/sirius/

`RestrictionDefinition`). Seguindo o cenário de exemplo, define-se um objeto contexto `Area`, com seus respectivos objetos `Coordinate`, para delimitar as coordenadas geográficas da Universidade Federal. Cria-se uma restrição `Speed` com velocidade máxima de 30 km/h e é feita uma associação entre contexto e restrição através de uma classe `Rule`. Por fim, cria-se uma condição temporal do tipo `WeekTimeCondition`, de acordo com os parâmetros de dias da semana, data e hora desejados. Os valores das coordenadas geográficas, da velocidade, os dias da semana, data e hora são definidos na forma textual.

Ainda de acordo com a Figura 4.6, a janela da ferramenta é dividida basicamente em quatro partes:

1. Área de criação dos modelos
2. Paleta de elementos de modelagem, dividida em
   - *Contexts*: elementos que representam os contextos, ou seja, `ContextDefinition`, `MU`, `Group`, `MUType`, `CircularArea`, `RectangularArea` e `PolygonalArea`;
   - *Restrictions*: elementos que representam as restrições, ou seja, `RestrictionDefinition`, `Speed`, `Path`, `MaxDistance`, `MinDistance`, `Access`, `Permanence` e `Punctuality`;
   - *Coordinates*: elementos que representam as coordenadas, ou seja, `Coordinate` e `CoordinateInTime`;
   - *Rules*: elementos que representam as regras, seus relacionamentos com contextos e restrições, além dos tipos de restrições temporais, ou seja, `Rule`, `Set Restriction`, `Set Context`, `WeekTime Condition` e `Date Condition`.
3. Campos de edição textual de valores, que servem para complementar a notação gráfica;
4. Tabela de visualização da formação das regras, que mostra em formato tabular todos os contextos, restrições e condições temporais que as formam.

Para criar um objeto na área do diagrama (parte 1), seleciona-se o ícone correspondente na paleta de elementos (parte 2). Uma vez criado, o objeto pode ser

livremente posicionado no diagrama, conforme melhor disposição visual ao usuário. Ao clicar em um objeto criado, seus atributos (descrição, identificador e outros parâmetros) são mostrados na parte inferior esquerda da interface (parte 3), para que sejam editados manualmente. Na parte inferior direita é mostrada de forma tabular a estrutura de formação da classe `Rule` criada (parte 4). Esta última parte não é editável, sendo apenas um reflexo da modelagem feita no diagrama (parte 1).

A criação dos objetos no diagrama segue todas as regras de modelagem definidas no metamodelo (Figura 4.2), segundo a linguagem MobCons-SL. Para criar um objeto `Context` qualquer, é requerida antes a criação de um objeto `ContextDefinition`, assim como para crição de qualquer `Restriction` é requerida antes a criação de um `RestrictionDefinition`. O objeto `Group`, especialmente, requer a criação de objetos `MU` para compô-lo, sendo tal associação feita em campo específico, chamado `Mu`, localizado na Parte 3. Os objetos `Coordinate` só podem ser associados aos objetos `RectangularArea`, `CircularArea`, `PolygonalArea` e `Path`, bem como os objetos `CoordinateInTime` só podem ser associados ao objeto `Punctuality`. Por fim, os objetos `WeekTimeCondition` e `DateCondition` só podem ser criados após criação de um objeto `Rule`, devendo ser a ele relacionados. Qualquer tentativa de criação de um objeto que fuja das regras de relacionamento estabelecidas no metamodelo será automaticamente negada pela linguagem MobCons-SL, garantindo assim a correta especificação, sem erros semânticos e de escrita.

Os relacionamentos dos objetos `Rule` com os objetos `Context` e `Restriction` são realizados através dos elementos `Set Context` e `Set Restriction`, respectivamente, localizados na paleta de elementos (parte 2) e devendo ser feitos da seguinte forma:

- `Rule` x `Context`: seleciona-se o objeto `Set Context`, clica-se no objeto `Rule` desejado e, em seguida, no objeto `Context`. Para relacionar mais de um contexto, repete-se a operação. É permitida a alteração ou exclusão de um contexto já relacionado, bastando para isso selecionar o objeto `Rule` e editar campo específico, chamado `Context`, localizado na Parte 3;

- `Rule` x `Restriction`: seleciona-se o objeto `Set Restriction`, clica-se no objeto `Rule` desejado e, em seguida, no objeto `Restriction`. É permitida

a alteração ou exclusão de uma restrição já relacionada, bastando para isso selecionar o objeto `Rule` e editar campo específico, chamado `Restriction`, localizado na Parte 3;

Conforme citado anteriormente, as regras de relacionamento são validadas automaticamente pela MobCons-SL. Além disso, a interface também permite a validação do modelo criado no tocante a outros pontos (como cardinalidade de um relacionamento), garantindo ainda mais a conformidade necessária entre a sintaxe concreta e abstrata. Essa validação é feita através da ação de um click com o botão direito do mouse, opção *"**Validate diagram**"*. A MobCons-AT emite uma mensagem informando o sucesso ou não da especificação modelada, conforme visto no trecho em destaque na Figura 4.7. No exemplo abaixo, tentou-se criar um objeto `RectangularArea` com apenas um objeto `Coordinate` e a ferramenta avisa da necessidade de no mínimo dois objetos `Coordinate`, conforme definido no metamodelo.

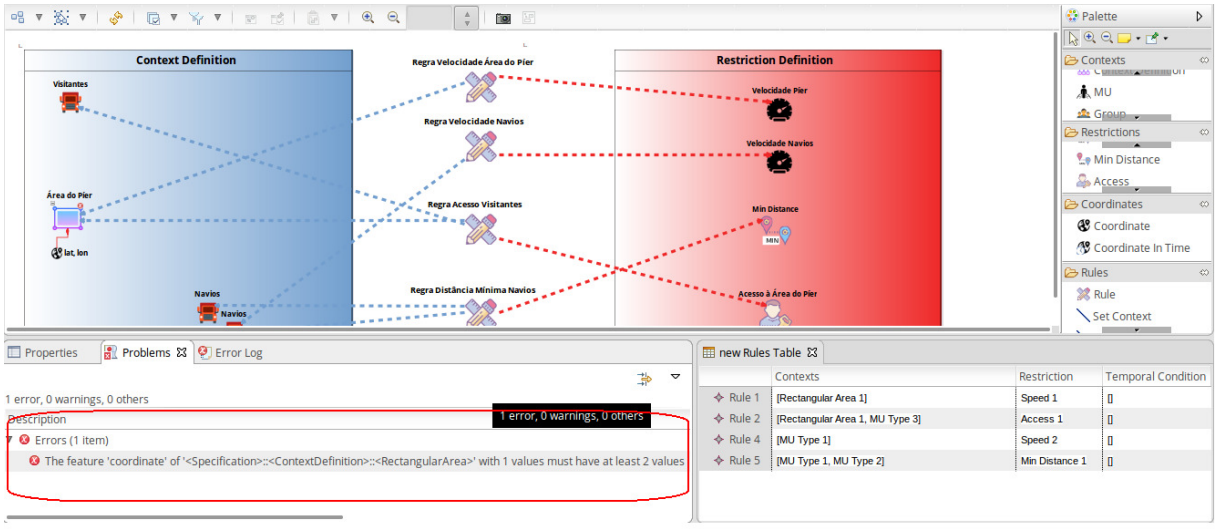

Figura 4.7: MobCons-AT: Mensagem de erro após validação do diagram.

4.5 Síntese

Neste capítulo foram abordados diversos aspectos da ferramenta MobCons-AT proposta, bem como sua DSML, o processo de transformação para geração de artefatos e o protótipo desenvolvido. Foram mostradas as motivações e os requisitos

para o seu desenvolvimento e descreveu-se também a sua arquitetura e suas principais funcionalidades.

Foi visto que a MobCons-AT tem o objetivo de permitir a especificação de restrições de mobilidade em alto nível pelo usuário final, por meio de uma DSML, chamada MobCons-SL. As especificações modeladas via MobCons-SL criam regras que passam por um processo de transformação M2T e geram artefatos responsáveis por acessar as classes implementadas na biblioteca `MobCons` que, por sua vez, instancia as regras CEP responsáveis pela leitura em tempo real dos fluxos de eventos contendo os dados de contexto dos dispositivos móveis. A arquitetura geral da MobCons-AT pode ser visualizada na Figura 4.8.

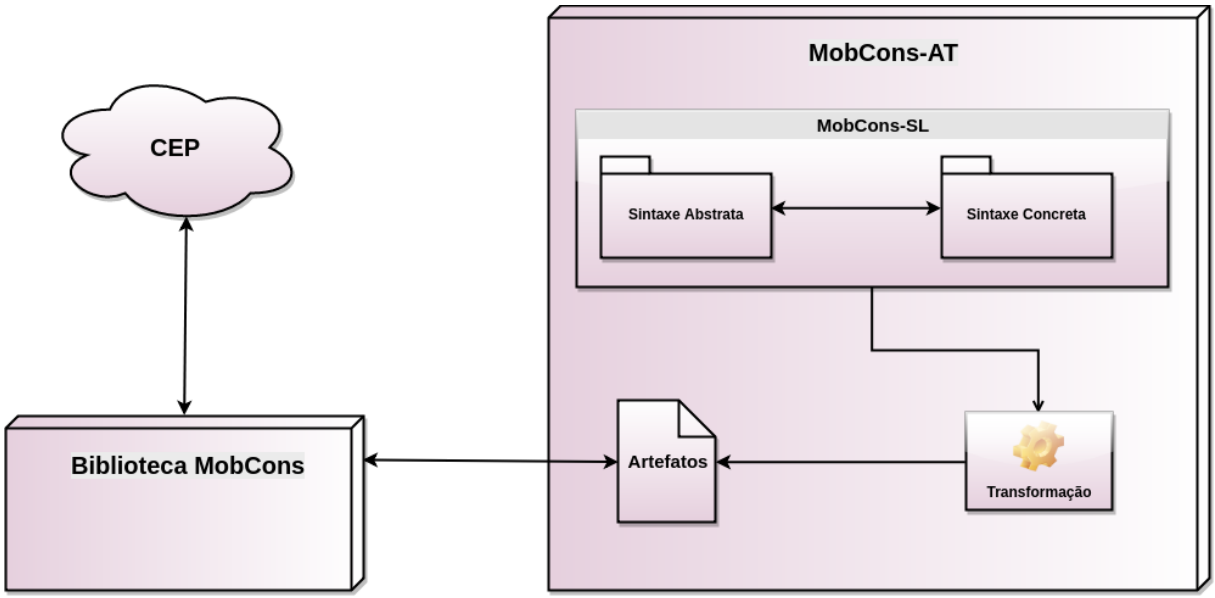

Figura 4.8: MobCons-AT: Arquitetura Geral

Foram apresentadas as classes que compõem a sintaxe abstrata, bem como suas representações gráfica na sintaxe concreta. Foi visto como o processo de transformação trabalha com o modelo gerado e produz os artefatos finais. Além disso, foi apresentado o protótipo da interface gráfica que representa visualmente todos os conceitos e regras de modelagem definidos no metamodelo, cabendo ao usuário final somente obter o conhecimento do domínio, o aprendizado das visualizações fornecidas e da forma como navegar entre elas.

5 Estudos de Casos

Esta seção tem como objetivo mostrar se a linguagem MobCons-SL, através da ferramenta MobCons-AT, é capaz de modelar com eficiência restrições de mobilidade de dispositivos móveis. Deseja-se verificar, portanto, se o usuário final será capaz de modelar graficamente os cenários apresentados e se a ferramenta produzirá artefatos que sejam capazes de instanciar as regras CEP responsáveis pelo monitoramento desejado. Para isso, foram feitos 5 (cinco) estudos de casos em diferentes cenários (empresa mineradora, transporte público, segurança pública, porto marítimo e aeroporto) com o objetivo de mostrar a utilização de diversos contextos, restrições e condições temporais, assim como a forma que eles se enquadrariam em situações do mundo real.

Em cada cenário é mostrada apenas uma possível modelagem, pois de acordo com o grau de conhecimento do domínio e da própria linguagem/ferramenta por parte do usuário, algumas maneiras diferentes para modelar um mesmo cenário seriam possíveis. Modelagens diferentes geram, obviamente, artefatos diferentes, mas desde que tenham sido modelados corretamente, isto é, tenham utilizado os contextos, restrições e condições temporais ideais ao caso, não devem alterar o objetivo final de instanciação das regras CEP e o monitoramento dos dispositivos móveis da forma esperada.

Na Seção Apêndice são mostrados os artefatos gerados para cada cenário tomado como estudo de caso, após modelagem gráfica e realização do processo de transformação.

5.1 Mineradora

Uma empresa mineradora quer limitar a velocidade máxima dos veículos dentro de seu pátio de minério de ferro a 40 km/h. Essa restrição de velocidade deve ser realizada entre as 07h e as 12h, de segunda a sexta-feira. Os veículos da empresa terceirizada Fórmula 1 não

devem exceder 60 km/h em todas as dependências da empresa. Além disso, os veículos do tipo escavadeira devem permanecer restritos ao pátio de minério de ferro.

5.1.1 Modelagem

A Figura 5.1 mostra uma possível modelagem para especificar as restrições do cenário proposto. Para tal, foram criados os seguintes objetos:

- Contextos
  - `PolygonalArea` (1 objeto): área geogáfica referente ao pátio de minérios, composta de quatro coordenadas fictícias;
  - `MU` (5 objetos): veículos pertencentes à empresa Fórmula 1;
  - `Group` (1 objeto): grupo formado pelos veículos da empresa Fórmula 1;
  - `MUType` (1 objeto): veículo tipo "ESCAVADEIRA".
- Restrições
  - `Speed` (2 objetos): velocidades máximas dentro do pátio de minério e dos veículos da empresa Fórmula 1;
  - `Permanence` (1 objeto): obrigatoriedade dos veículos tipo "ESCAVADEIRA"permanecerem dentro do pátio de minérios.
- Condição Temporal
  - `WeekTimeCondition` (1 objeto): janela de tempo para o limite de velocidade de veículos dentro do pátio de minérios.

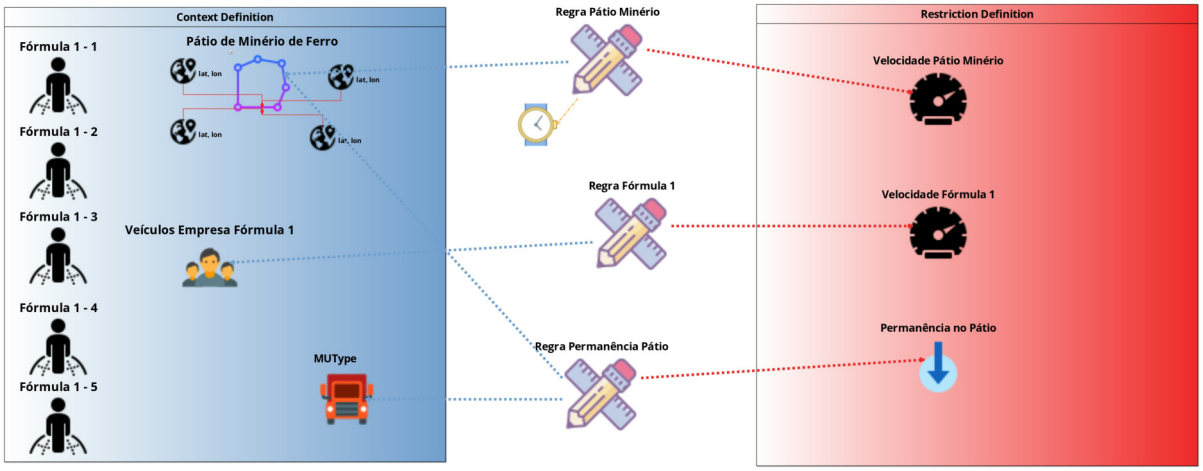

Figura 5.1: MobCons-AT: Modelagem de Cenário Mineradora

O artefato gerado pela modelagem definida para este cenário, após processo de transformação, encontra-se disponível no Apêndice A.

5.1.2 Discussão

A área referente ao pátio de minérios foi criada como poligonal, mas poderia ser definida de qualquer tipo. A escolha do tipo de área deve levar em conta o grau de precisão que se deseja obter na definição dos limites geográficos. As coordenadas geográficas do mundo real precisam ser obtidas através de um algum aplicativo que trabalhe com mapas e forneça tais dados, como o Google Maps[9], pois a MobCons-AT ainda não fornece essas informações em sua interface.

Os veículos da empresa terceirizada podem ser modelados individualmente, via objeto `MU`, ou através da criação de um grupo, via objeto `Group`, ou ainda via objeto `MUType`. Na primeira abordagem, obviamente, quanto maior o número de veículos, mais objetos precisam ser definidos graficamente, o que pode tornar a visualização não muito agradável. Na segunda abordagem, é importante observar que todas os veículos precisam ser definidos individualmente, pelo objeto `MU`, antes de serem incluídas em um grupo. Um grande número de veículos acarretaria em um grande número de objetos na interface gráfica. Na terceira abordagem, um único objeto `MUType` seria necessário, atribuindo aos veículos da empresa um tipo

[9]Google Maps:https://www.google.com.br/maps

"FORMULA1". Como já explicado na Seção 4.2.1, o tipo de dispositivo vem no fluxo de evento e assim é facilmente capturado.

A restrição `Permanence` requer a definição de um contexto de área, juntamente com outro contexto móvel, seja `MU`, `Group` ou `MUType`. Neste caso, criou-se um objeto `MUType` que abrange de uma só vez todos os veículos do tipo "ESCAVADEIRA", especificando que esses veículos devem permanecer nessa área. Quando se deseja atribuir essa restrição a apenas um dispositivo móvel, utiliza-se `MU`, mas quando a restrição deve ser imposta a dois ou mais dispositivos, os objetos `Group` e `MUType` devem ser utilizados. O mesmo vale para a reatrição `Access`.

Para um cenário um tanto complexo quanto este, é exigido um bom conhecimento, tanto do domínio do problema quanto da ferramenta/linguagem. Usuários menos experientes podem sentir um pouco de dificuldade no início.

5.2 Transporte Público

Uma empresa de transporte deseja controlar o deslocamento de sua frota de ônibus da seguinte forma: cada ônibus deve seguir uma rota predefinida, além de manter o limite de velocidade de 60 km/h. Os ônibus devem manter uma distância mínima de aproximadamente 3 km entre si, na sequência em que saem da garagem, a fim de garantir a uniformidade na circulação.

5.2.1 Modelagem

A Figura 5.2 mostra uma possível modelagem para especificar as restrições do cenário proposto. Para tal, foram criados os seguintes objetos:

- Contextos
  - `MU` (4 objetos): ônibus pertencentes à frota da empresa de transporte público.
- Restrições
  - `Speed` (1 objeto): velocidade máxima da frota de ônibus;
  - `MinDistance` (1 objeto): distância mínima entre os ônibus da frota;

– `Path` (1 objeto): rota obrigatória a ser seguida pelos ônibus.

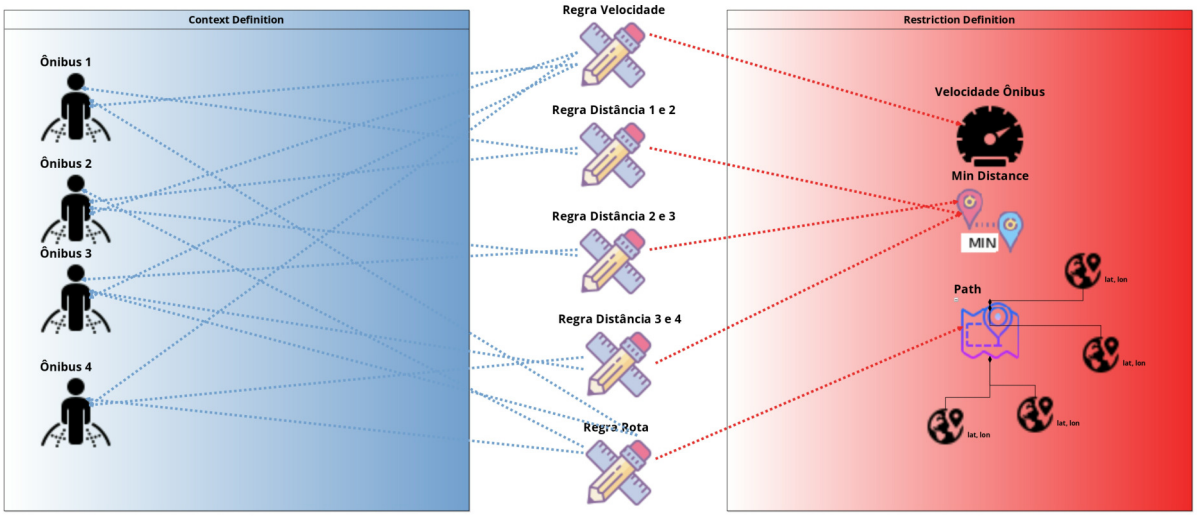

Figura 5.2: MobCons-AT: Modelagem de Cenário Transporte Público

O artefato gerado pela modelagem definida para este cenário, após processo de transformação, encontra-se disponível no Apêndice B.

5.2.2 Discussão

A restrição `MinDistance` exigiu a inclusão de todos os ônibus da frota como contextos, uma vez que ela, por definição, é a distância mínima entre dois pontos. Nesse caso, foi preciso uma regra para cada grupo de dois ônibus. Supondo que a sequência de saída da garagem seja: Ônibus 1 → Ônibus 2 → Ônibus 3 → Ônibus 4, foram necessárias três Regras unindo (Ônibus 1, Ônibus 2), (Ônibus 2, Ônibus 3) e (Ônibus 3, Ônibus 4). Uma outra forma de modelagem desta restrição, bem mais simples, seria através da definição de dois contextos `Group` contendo os mesmos objetos `MU` ou através de dois contextos `MUType` para os mesmos tipos ("Ônibus"). Internamente, a biblioteca `MobCons` consegue definir a distância aos membros dessas classes.

É importante observar que a restrição `MinDistance`, assim como a `MaxDistance`, para casos de um grande número de objetos, exige um grande número de classes `Rule` de modo a agrupá-los dois a dois, o que aumenta um pouco mais a complexidade da modelagem. Uma forma de diminuir o grau de complexidade seria agrupar os dispositivos em contextos `Group` ou `MUType`.

5.3 Segurança Pública

O Departamento de Segurança Pública quer controlar o uso de suas viaturas policiais. Seguindo as regras do Secretariado, cada viatura deve cobrir uma área predeterminada, dentro de um período de tempo específico, que deve ser das 06h às 18h, de segunda a sexta, e das 10h às 22h, aos sábados e domingos. As áreas de atuação das viaturas deverão ser: Itaqui-Bacanga, Centro e Praias. Com exceção dos momentos em que são chamados para emergências ou flagrantes delitos, as viaturas devem fazer rondas apenas na sua área de operação, a uma velocidade máxima de 40 km/h.

5.3.1 Modelagem

A Figura 5.3 mostra uma possível modelagem para especificar as restrições do cenário proposto. Para tal, foram criados os seguintes objetos:

- Contextos
  - `MU` (6 objetos): viaturas pertencentes à frota da segurança pública;
  - `CircularArea` (3 objetos): áreas de atuação das viaturas;
  - `Group` (3 objetos): grupos de viaturas, por área de atuação.
- Restrições
  - `Speed` (1 objeto): velocidade máxima da frota de viaturas, quando não estiverem em emergência ou flagrante delito;
  - `Permanence` (1 objeto): obrigatoriedade das viaturas permanecerem dentro das suas áreas de atuação.
- Condição Temporal
  - `WeekTimeCondition` (6 objetos): janelas de tempo para a permanência das viaturas em suas respectivas áreas de atuação.

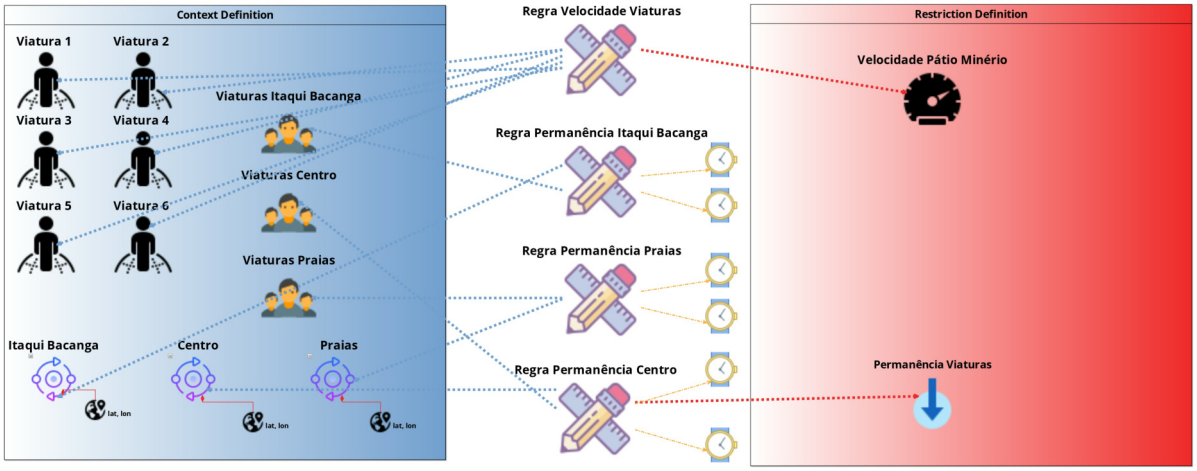

Figura 5.3: MobCons-AT: Modelagem de Cenário Segurança Pública

O artefato gerado pela modelagem definida para este cenário, após processo de transformação, encontra-se disponível no Apêndice C.

5.3.2 Discussão

Tal qual descrito no cenário 5.1, o contexto de área poderia ser definido de qualquer tipo. Neste caso, optou-se por uma área circular. Cada área de atuação das viaturas (Itaqui-Bacanga, centro e praias) foi definida por um objeto `CircularArea` distinto.

Para definir que viaturas fazem parte de que área de atuação, utilizou-se objetos `Group`, um para cada área. É importante observar que todas as viaturas precisam ser definidas individualmente, pelo objeto `MU`, antes de serem incluídas em um grupo. Um grande número de viaturas acarretaria em um grande número de objetos na interface gráfica. Mais uma vez, uma outra alteranativa interessante seria a utilização de objetos `MUType`, um para cada área de atuação. Por exemplo, poderiam ser criados três tipos: "VIATURAS_ITAQUI_BACANGA", "VIATURAS_CENTRO"e "VIATURAS_PRAIAS". Nesta abordagem, apenas três objetos modelariam todas as viaturas, independente do número existente.

Para a restrição `Permanence`, foram criadas três regras associando cada grupo de viaturas com suas áreas de atuação, indicando assim que as viaturas devem permanecer dentro de suas áreas. Referentes a essa permanência, o cenário requer ainda duas condições temporais (das 06h às 18h, de segunda a sexta, e das 10h às

22h, aos sábados e domingos). Da forma como a interface gráfica foi implementada, deve-se arrastar o objeto `WeekTimeCondition` até o objeto `Rule` correspondente. Por esse motivo, foi preciso criar objetos `WeekTimeCondition` iguais para cada classe `Rule`, ou seja, para cada regra de permanência, foram criadas as mesmas condições temporais utilizando-se objetos diferentes, mas com os mesmos valores, o que é um tanto redundante, embora não cause erros. Uma forma mais elegante de implementação da interface seria criar somente dois objetos `WeekTimeCondition` e reutilizá-los em todas as regras, já que o metamodelo permite.

5.4 Porto Marítimo

No Porto de Itaqui, por motivos de segurança, somente funcionários autorizados da área operacional e veículos de carga e descarga podem transitar pelos piers, cuja velocidade máxima para circulação de veículos é de 10 km/h. Os navios, quando ancorados no mar aguardando atracar, devem manter uma distância mínima de 1 km uns dos outros e ao se deslocarem para atracamento no píer, devem manter uma velocidade mínima de 3 km/h e máxima de 10 km/h.

5.4.1 Modelagem

A Figura 5.4 mostra uma possível modelagem para especificar as restrições do cenário proposto. Para tal, foram criados os seguintes objetos:

- Contextos
  - `MU` (3 objetos): visitantes do Porto e navios;
  - `RectangularArea` (1 objeto): área do píer;
- Restrições
  - `Speed` (2 objetos): velocidades máximas de qualquer veículo na área do píer e dos navios durante manobra de atracamento;
  - `Access` (1 objeto): proibição de acesso para visitantes acessarem a área do píer;

– `MinDistance` (1 objeto): distância mínima entre navios enquanto aguardam para atracar.

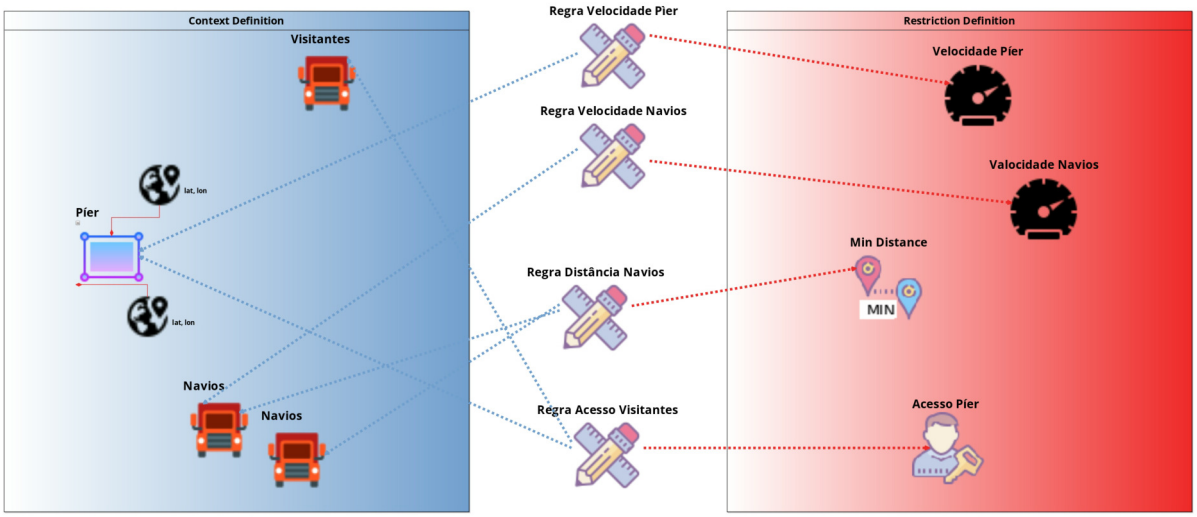

Figura 5.4: MobCons-AT: Modelagem de Cenário Porto Marítimo

O artefato gerado pela modelagem definida para este cenário, após processo de transformação, encontra-se disponível no Apêndice D.

5.4.2 Discussão

Tal qual descrito no Cenário 5.1, o contexto de área poderia ser definido de qualquer tipo. Neste caso, optou-se por uma área retangular.

A restrição `Access` define qual contexto não pode acessar uma área definida. Como a permissão é dada somente para funcionários operacionais e veículos de carga, restringiu-se o acesso a visitantes, por exemplo. De acordo com a necessidade, outros contextos cujo acesso ao píer fosse proibido poderiam ser incluídos na restrição. Em vez de definir todos os visitantes individualmente através do contexto `MU`, o que seria inviável, optou-se por defini-los com o contexto `MUType`, o que torna a modelagem mais simples e abrange um número indefinido de pessoas.

Este cenário, embora aparentemente simples, mostrou o uso de um contexto interessante, o `MUType`, no que diz respeito à especificação da restrição de `MinDistance)` aos navios. Conforme já foi dito anteriormente na Seção 4.2.1, essa restrição (assim como a `MaxDistance`) requer a existência de dois contextos. Uma

possibilidade de modelagem foi mostrada no Cenário 5.2, onde os ônibus foram agrupados de dois a dois. Já neste exemplo, criou-se um contexto com a classe `MUType` definindo todos os objetos do tipo "Navio". Para atender à assinatura da restrição de `MinDistance`, foi criado outro objeto `MUType` do tipo "Navio". A biblioteca `MobCons` entende que a restrição se refere a todos os objetos desse tipo e consegue implementar essa condição nas consultas CEP.

5.5 Aeroporto

O Aeroporto Internacional local quer limitar o movimento de pessoas em algumas áreas de segurança, por exemplo, na área de manobras de aeronaves, onde apenas circulam funcionários autorizados e veículos que transportam bagagens, mantendo a velocidade máxima de 20 km/h. Em dias específicos, como feriado de Natal e Ano Novo, essa velocidade cai para 10 km/h, devido ao fluxo mais intenso de aeronaves na pista.

5.5.1 Modelagem

A Figura 5.5 mostra uma possível modelagem para especificar as restrições do cenário proposto. Para tal, foram criados os seguintes objetos:

- Contextos
  - `RectangularArea` (1 objeto): área de manobras das aeronaves;
  - `MUType` (2 objetos): passageiros e veículos transportadores de bagagens.
- Restrições
  - `Speed` (2 objetos): velocidades máximas na área de manobras em dias normais e em dias especiais (Natal e Ano Novo);
  - `Access` (1 objeto): proibição de acesso para passageiros e veículos transportadores de bagagens acessarem a área de manobras de aeronaves.

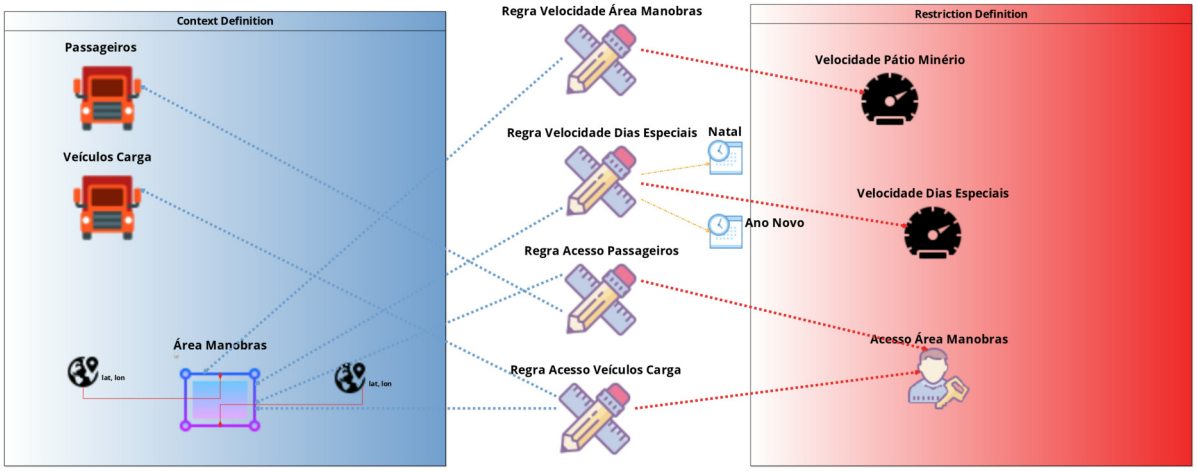

Figura 5.5: MobCons-AT: Modelagem de Cenário Aeroporto

O artefato gerado pela modelagem definida para este cenário, após processo de transformação, encontra-se disponível no Apêndice E.

5.5.2 Discussão

Tal qual descrito no Cenário 5.1, o contexto de área poderia ser definido de qualquer tipo. Neste caso, optou-se por uma área retangular.

A restrição `Access` define qual contexto não pode acessar uma área definida. Como a permissão é dada somente para funcionários autorizados e veículos transportadores de bagagens, restringiu-se o acesso a passageiros e veículos de carga, por exemplo. De acordo com a necessidade, outros contextos poderiam ser incluídos na restrição. Em vez de definir todos os passageiros e veículos de carga individualmente através do contexto `MU`, o que seria inviável, optou-se por defini-los com o contexto `MUType`, um para cada tipo, o que torna a modelagem mais simples.

A condição temporal de velocidade em dias especiais (Natal e Ano Novo) foi definida pelo uso da classe `DateCondition` na forma "dd/mm"(dia/mês), sem especificar o ano. Desta forma, a restrição fica válida nesses dias específicos, em todos os anos subsequentes.

5.6 Validação de Eficácia

Esta seção tem por objetivo testar se os artefatos gerados pela ferramenta, após processo de transformação, conseguem de fato instanciar as regras CEP responsáveis pelo monitoramento desejado. Para tanto, foi escolhido o artefato gerado pelo cenário 5.5, disponível no Apêndice E. O teste consistiu em adicionar ao artefato escolhido algumas linhas de código simulando exemplos de fluxos de eventos CEP preparados tanto para quebrar as regras especificadas, quanto para não quebrá-las. Ao final, a `MobCons` gerou um log de execução através do qual foi constatado o sucesso do teste. O trecho de código abaixo mostra os eventos criados para teste.

```
1  //1-Entra na área, tem permissão de acesso e não excedeu velocidade
2  MobilityEvent mampEvt = new MobilityEvent(2040, 150, 150, 10, 0, "Veiculo");
3  System.out.println(mampEvt);
4  mobconsCore.handleEvent(mampEvt);
5  //2-Entra na área, tem permissão de acesso e excedeu velocidade
6  mampEvt = new MobilityEvent(2040, 150, 150, 30, 200, "Veiculo");
7  System.out.println(mampEvt);
8  mobconsCore.handleEvent(mampEvt);
9  //3-Não entra na área e excedeu velocidade
10 mampEvt = new MobilityEvent(2040, 250, 250, 50, 300, "Veiculo");
11 System.out.println(mampEvt);
12 mobconsCore.handleEvent(mampEvt);
13 //4-Entra na área, não é autorizado e excedeu velocidade
14 mampEvt = new MobilityEvent(2040, 150, 150, 50, 300, "Veiculos Carga");
15 System.out.println(mampEvt);
16 mobconsCore.handleEvent(mampEvt);
```

Listing 2: Artefato gerado após transformação, baseado no exemplo da Seção 4.2.1.

A classe `MobilityEvent` utilizada no código acima representa o fluxo de eventos CEP que é gerado a partir dos dispositivos móveis e lido pela `MobCons`. Como assinatura, recebe os seguintes parâmetros, respectivamente:

1. `muId`: código identificador do dispositivo móvel;
2. `latitude`: valor da coordenada geográfica latitude onde o dispositivo móvel se encontra;

3. `longitude`: valor da coordenada geográfica longitude onde o dispositivo móvel se encontra;

4. `speed`: valor da velocidade do dispositivo móvel;

5. `timestamp`: marco temporal da ocorrência do evento;

6. `kindOfMu`: tipo de dispositivo móvel.

A Figura 5.6 mostra o log gerado pela biblioteca `MobCons`, quando da execução do teste. Na parte 1, pode-se obervar todas as consultas CEP geradas pelas regras especificadas no artefato, via linguagem da MobCons-AT. No log, as mensagens de violação das regras são delimitadas pela mensagem *"Constraint Violation detected!"*, encontrada nas partes 3 e 5, informando qual restrição foi violada e qual o fluxo de evento que causou a violação. A seguir são explicados os resultados dos processamentos de cada fluxo de evento.

```
select * from MobilityEvent.std:lastevent() as evt where evt.speed>20.0 and (evt.point.x in [100.0 : 200.0]  and
evt.point.y in [100.0 : 200.0])

select * from MobilityEvent.std:lastevent() as evt where (evt.point.x in [100.0 : 200.0]  and evt.point.y in
[100.0 : 200.0]) and evt.kindOfMU="Passageiros"

select * from MobilityEvent.std:lastevent() as evt where (evt.point.x in [100.0 : 200.0]  and evt.point.y in
[100.0 : 200.0]) and evt.kindOfMU="Veiculos Carga"

select * from MobilityEvent.std:lastevent() as evt where evt.speed>10.0 and (evt.point.x in [100.0 : 200.0]  and
 evt.point.y in [100.0 : 200.0]) and (evt.timestamp.getDayOfMonth() = 31 and evt.timestamp.getMonthOfYear() = 12)

select * from MobilityEvent.std:lastevent() as evt where evt.speed>10.0 and (evt.point.x in [100.0 : 200.0]  and
 evt.point.y in [100.0 : 200.0]) and (evt.timestamp.getDayOfMonth() = 25 and evt.timestamp.getMonthOfYear() = 12)
```

1

```
MobilityEvent{muID=2040, point=Point2D.Double[150.0, 150.0], speed=10.0, timestamp=0, kindOfMU='Veiculo'}
MobilityEvent{muID=2040, point=Point2D.Double[150.0, 150.0], speed=30.0, timestamp=200, kindOfMU='Veiculo'}
```

2

```
*------- Constraint Violation detected! [start] -------*
Restriction: SpeedRestriction{maxSpeed=20.0, minSpeed=-1.0}
Event[0]: MobilityEvent{muID=2040, point=Point2D.Double[150.0, 150.0], speed=30.0, timestamp=200, kindOfMU='Veiculo'}
*------- Constraint Violation detected! [end] -------*
```

3

```
MobilityEvent{muID=2040, point=Point2D.Double[250.0, 250.0], speed=50.0, timestamp=300, kindOfMU='Veiculo'}
MobilityEvent{muID=2040, point=Point2D.Double[150.0, 150.0], speed=50.0, timestamp=300, kindOfMU='Veiculos Carga'}
```

4

```
*------- Constraint Violation detected! [start] -------*
Restriction: SpeedRestriction{maxSpeed=20.0, minSpeed=-1.0}
Event[0]: MobilityEvent{muID=2040, point=Point2D.Double[150.0, 150.0], speed=50.0, timestamp=300, kindOfMU='Veiculos Carga'}
*------- Constraint Violation detected! [end] -------*

*------- Constraint Violation detected! [start] -------*
Restriction: AreaRestriction{inside=false, linearRing=[Point2D.Double[100.0, 100.0], Point2D.Double[200.0, 200.0]], radius=-1
Event[0]: MobilityEvent{muID=2040, point=Point2D.Double[150.0, 150.0], speed=50.0, timestamp=300, kindOfMU='Veiculos Carga'}
*------- Constraint Violation detected! [end] -------*
```

5

Figura 5.6: Log gerado pela biblioteca `MobCons`, após execução de artefato gerado.

No fluxo de evento 1 do trecho de código de teste (linha 2), foi definido um dispositivo móvel (id = 2040), tipo "Veículo", que entrou na área de manobras (coordenadas geográficas se encontram no interior das coordenadas delimitadas no cenário), a uma velocidade de 10 km/h. De acordo com as restrições especificadas, o dispositivo acessou a área, tem permissão de acesso e não excedeu o limite de

velocidade máxima da área. Portanto, **não quebrou nenhuma regra**, conforme pode ser comprovado no log, na parte 2 da Figura 5.6.

No fluxo de evento 2 (linha 6), o dispositivo móvel, tipo "Veículo", entrou na área de manobras, desta vez a uma velocidade de 30 km/h. Neste caso, **houve a violação da regra especificada no tocante à velocidade máxima**, que devia ser de apenas 20 km/h, conforme comprovado no log, na parte 3 da Figura 5.6.

No fluxo de evento 3 (linha 10), o dispositivo móvel, tipo "Veículo", não entrou na área de manobras, a uma velocidade de 50 km/h. Neste caso, **não houve a violação da regra especificada no tocante à velocidade máxima**, uma vez que a mesma só é definida para o interior da área de manobras, conforme comprovado no log, na parte 4 da Figura 5.6.

No fluxo de evento 4 (linha 14), o dispositivo móvel, tipo "Veículos Carga", entrou na área de manobras, a uma velocidade de 50 km/h. Neste caso, **houve dupla violação das regras especificadas no tocante à velocidade máxima e ao acesso** (proibido para veículos de carga), conforme comprovado no log, na parte 5 da Figura 5.6.

Embora tenha-se obtido sucesso ao testar o cenário escolhido, a realização de testes de eficácia mais abrangentes, com mais cenários e situações, seria ideal para buscar obter uma comprovação da infalibilidade do processo como um todo, desde a modelagem por parte do usuário até a geração do artefato.

5.7 Síntese

Neste capítulo foram apresentados cinco estudos de casos onde a linguagem MobCons-SL foi utilizada, via interface gráfica da MobCons-AT. Através de cada cenário, mostrou-se que o uso da ferramenta foi motivado pela necessidade de especificar restrições de mobilidade a dispositivos móveis em alto nível, dispensando o desenvolvimento em linhas de código por parte de programadores, cabendo ao usuário final unicamente a tarefa de modelagem.

Mostrou-se que a ferramenta foi capaz de modelar os cenários apresentados e, por conseguinte, se mostra aberta a modelagens de quaisquer outros cenários que apresentem domínio semelhante.

Observou-se que há casos com mais de uma possibilidade de modelagem (por exemplo, quanto ao uso dos contextos, onde uma área geográfica poderia ser modelada da forma circular, retangular ou poligonal, dependendo da precisão que se quer obter), cujo grau de facilidade depende do nível de conhecimento por parte do usuário, tanto em relação ao domínio quanto ao uso da ferramenta. Da mesma forma, algumas restrições mais complexas, tais como `Access`, `Permanence`, `MaxDistance` e `MinDistance`, requerem bom conhecimento por parte do usuário, a fim de que sejam corretamente utilizadas.

Observou-se que alguns pontos ainda precisam ser melhorados, tais como a necessidade de uma aplicação externa que faça uso de mapas para obter coordenadas geográficas reais e a impossibilidade de reutilização de um mesmo objeto de condição temporal para várias regras (limitação esta imposta pela forma como a interface gráfica foi projetada), pois em cenários com uso de uma mesma condição temporal associada a várias regras, ocorrerá redundância desse objeto.

Por fim, através de um teste de eficácia, mostrou-se que a MobCons-AT conseguiu gerar um artefato capaz de instanciar corretamente as regras CEP, através da bibioteca `MobCons`. O artefato gerado no cenário 5.5 foi testado com êxito na `MobCons`, abordando situações onde as regras especificadas poderiam ser quebradas ou não. Mas embora tenha-se obtido sucesso, testes mais abrangentes devem ser feitos para se concluir a infalibiidade do processo como um todo.

6 Experimento para Avaliar a MobCons-AT

A experimentação é a forma mais confiável de verificar teorias a fim de que elas possam ser comprovadas e corrigidas [50], além de ser um processo sistemático que envolve algumas etapas como observação do problema, formulação das questões de pesquisa, validação de hipóteses, execução do experimento, análises para estabelecer relações e, finalmente, formular conclusões [40, 49].

São encontrados na literatura alguns trabalhos que oferecem soluções que permitem o monitoramento de dispositivos móveis, sem implementar uma linguagem de modelagem específica para o domínio, sendo aqueles julgados mais relevantes apresentados no Capítulo 3. Para comprovar a eficácia da abordagem utilizando uma linguagem de domínio, este capítulo apresenta um estudo experimental que visa analisar o suporte à especificação de restrições de mobilidade a dispositivos móveis utilizando a linguagem MobCons-SL, implementada para uso através da ferramenta de autoria MobCons-AT, apresentadas no Capítulo 4.

Todas as seções deste capítulo, se concentram em descrever o estudo experimental, sendo organizadas da seguinte forma: a Seção 6.1 detalha o objetivo pricipal deste experimento. A Seção 6.2 mostra as questões e métricas abordadas. A Seção 6.3 aborda as variáveis consideradas no processo. A Seção 6.4 detalha os objetos de estudo utilizados no experimento. A Seção 6.5 mostra como se deu a seleção dos participantes. A Seção 6.6, por sua vez, aborda o contexto e a instrumentação do experimento. A Seção 6.7 mostra como se deu a elaboração do projeto do experimento como um todo. A Seção 6.8 explica em detalhes todas as fases de operação do experimento. A Seção 6.9 mostra os resultados obtidos no experimento. A Seção 6.10 aborda a confiabilidade do experimento, mostrando os aspectos relacionados à validade. Por fim, a Seção 6.11 faz uma análise e traz discussões sobre o processo.

6.1 Objetivo

O objetivo do experimento é analisar o suporte à especificação de restrições de mobilidade por parte de usuários finais, especialistas ou não, leigos ou não em desenvolvimento de *software*, avaliando o quão intuitiva e eficaz é a solução criada. Sendo assim, será avaliado o efeito desse suporte com respeito ao esforço e à percepção da qualidade na tarefa de especificação. Para tanto, o experimento contará com a participação de analistas e técnicos em tecnologia da informação e outras áreas acadêmicas, estudantes de graduação e estudantes de pós-graduação.

6.2 Questões e Métricas

A maioria das avaliações de usabilidade reúne dados quantitativos subjetivos e objetivos no contexto de cenários realistas de uso. Dados subjetivos são medidas das opiniões ou atitudes dos participantes em relação à sua percepção de usabilidade. Os dados objetivos são medidas do desempenho dos participantes (como o tempo de conclusão do cenário e a taxa de conclusão do cenário de sucesso) [31,32]. Com base nesta abordagem, este estudo foi planejado com o objetivo de responder às questões relacionadas ao esforço e à percepção da qualidade.

6.2.1 Esforço

Indica o tempo como métrica para especificação de restrições de forma correta, garantindo ao final do processo um bom resultado (infalibilidade). Utilizou-se a medida de tempo em minutos como métrica para avaliar o esforço aplicado na especificação de restrições de mobilidade. Quanto maior o tempo, maior o esforço aplicado. O objetivo desta métrica é servir como apoio adicional na análise da usabilidade e intuitividade da ferramenta, indicando o esforço a mais feito por um usuário ainda inexperiente na ferramenta, comparado a um usuário experiente.

Uma vez que a MobCons-AT ainda é um protótipo, utilizou-se como base de comparação a média em minutos para modelagens de cenários feitas por dois usuários com grande expertise do domínio de restrições de mobilidade e conhecimento da

ferramenta, abordando cenários com três níveis de complexidade, conforme descrito na Figura 6.1:

Tabela 6.1: Tempo médio para modelagem na MobCons-AT por especialistas

Nível	Descrição	Tempo Médio (mm:ss)
Fácil	Cenário com apenas um contexto e uma restrição, resultando em apenas uma regra.	01:00
Médio	Cenário com mais de um contexto e uma restrição, resultando em apenas uma regra com condição temporal.	01:30
Difícil	Cenário com mais de um contexto e mais de uma restrição, resultando em mais de uma regra.	04:00

A questão de pesquisa relacionada ao esforço foi:

O esforço necessário para um usuário final, sem conhecimento prévio do domínio e sem experiência na MobCons-AT, foi em média superior em quantos minutos ao esforço médio realizado por um especialista?

6.2.2 Qualidade

Indica o nível de satisfação dos voluntários quanto à usabilidade da ferramenta. Considerando a subjetividade que a natureza dessa questão levanta, questionários para aferir a satisfação dos voluntários foram utilizados. Objetiva-se a partir desse estudo analisar se a ferramenta é bem sucedida em estabelecer uma interface intuitiva, apresentando-se como um canal de fácil manipulação de especificação de restrições de mobilidade e fornecendo recursos suficientes para que tais especificações sejam programáveis em alto nível.

A questão de pesquisa relacionada à percepção da qualidade foi:

A percepção da qualidade do suporte oferecido aos usuários utilizando a MobCons-AT foi, em média, positiva, neutra ou negativa?

6.3 Variáveis

As variáveis consideradas neste experimento são as seguintes:

- independentes: São disponibilizadas como entrada para o experimento, podendo ser manipuladas durante o experimento. Neste experimento, utilizou-se a

IDE Eclipse Obeo Designer Community (versão 10.1) um pacote pronto para uso que inclui todas as ferramentas necessárias ao projeto em estudo, dentre as quais, destacam-se: o *plugin* Acceleo (versão 6.0), utilizado para suportar as transformações entre modelos, o *plugin* Sirius (versão 5.1), utilizado para suportar a notação gráfica desenvolvida para interface com usuário final, além de outras ferramentas do projeto Eclipse, tais como os *plugins* Eclipse Modeling Framework (EMF) e Graphical Modeling Framework (GMF).

- dependentes: São aquelas resultantes após a execução do experimento, oferecendo os resultados referentes aos tratamentos aplicados no experimento. As variáveis dependentes consideradas nesse experimento, para analisar o suporte às tarefas de especificação de restrições de mobilidade são o esforço e a percepção de qualidade dos participantes.

Configuração mínima necessária para o funcionamento da IDE:

- Processador 2 Ghz 64-bit (x64);
- 1 GB de memória de sistema disponível;
- Mínimo de 500 MB de espaço disponível no disco rígido para instação;
- Java Runtime Environment (JRE), versão 8.

6.4 Objetos de Estudo

Os objetos de estudo são os meios utilizados para verificar a relação de causa-efeito nas questões adotadas. Durante a execução do experimento, os tratamentos com o uso da MobCons-AT são aplicados aos objetos. Nesse experimento, considerou-se como objeto três cenários, com graus de dificuldade crescente, caracterizados como fácil, médio e difícil. Esse objeto foi escolhido para que o experimento não se tornasse muito longo, no tocante aos cenários fácil e médio, mas dificultasse um pouco a tarefa dos participantes, no tocante ao cenário difícil.

O detalhe dos cenários utilizados como objetos de estudo são mostrados na Tabela 6.2 abaixo:

Tabela 6.2: Cenários Utilizados como Objetos de Estudo

Nível	Descrição	Cenário
Fácil	Cenário com apenas um contexto e uma restrição, resultando em apenas uma regra.	"Todos as viaturas policiais do Estado do Maranhão devem transitar a uma velocidade máxima de 60 km/h".
Médio	Cenário com mais de um contexto e uma restrição, resultando em apenas uma regra com condição temporal.	"Uma empresa de transporte coletivo deseja que todos os seus veículos do tipo “ônibus” transitem no centro da cidade a uma velocidade máxima de 40 km/h, de segunda a sexta, entre 8h e 18h".
Difícil	Cenário com mais de um contexto e mais de uma restrição, resultando em mais de uma regra.	"No Porto de Itaqui, por motivos de segurança, somente funcionários autorizados da área operacional e veículos de carga e descarga podem transitar pelos piers, cuja velocidade máxima para circulação de veículos é de 10 km/h. Os navios, quando ancorados no mar aguardando atracar, devem manter uma distância mínima de 1 km uns dos outros e ao se deslocarem para atracamento no píer, devem manter uma velocidade mínima de 3 km/h e máxima de 10 km/h"

6.5 Seleção dos Participantes

A seleção dos participantes é uma fase importante para o experimento, porque os participantes escolhidos devem possibilitar uma generalização dos resultados obtidos. Assim como nos trabalhos de Oliveira et al. e Osvaldo Silva et al. [40, 49], a escolha dos participantes não foi feita de forma aleatória, como se espera de um experimento, mas conforme a disponibilidade encontrada. Esse tipo de amostragem é conhecida como amostragem de conveniência.

Os participantes alvos deste experimento são analistas e técnicos com formação profissional em Ciência da Computação ou áreas afins, que tenham ou não experiência em desenvolvimento de software, ou com o paradigma Mode-Driven Engeneering (MDE) ou IDE Eclipse. Para alcançar a generalização desejada, foram selecionados profissionais de outras áreas de formação, como Direito, Administração, Ciências Contábeis, Engenharia e Nutrição, além de alunos de graduação ou de pós-graduação de cursos em Ciência da Computação ou áreas afins que tenham ou não a experiência citada acima. Embora a amostragem não tenha sido tão grande em termos de número de participantes, procurou-se diversidade no grau de formação e nos perfis acadêmicos e profissionais por considerar-se que os voluntários com alguma experiência em ferramentas MDE ou IDE Eclipse tenham naturalmente mais facilidade na execução do experimento, se comparados com voluntários de outras áreas sem nenhuma experiência nesse tipo de aplicação.

Um questionário foi aplicado para caracterização dos participantes. O propósito foi descobrir o perfil de cada um em termos acadêmicos, profissionais e de conhecimento em desenvolvimento de software, MDE ou IDE Eclipse. Esse questionário pode ser encontrado na Seção Apêndice F.

No processo de seleção, 30 (trinta) questionários foram preenchidos, onde 100% dos participantes concluíram todas as etapas do experimento. Destes, 70% são profissionais que atuam no mercado (empresas privadas e servidores públicos) e 30% são apenas estudantes. Em termos de formação acadêmica, o perfil dos participantes foi equilibrado, sendo 60% da área de Computação e os 40% restantes composto de diversas áreas, tais como Administração, Direito, Química, Ciências Contábeis, Nutrição, Engenharia e técnicos. Quanto à titulação, o maior percentual foi de graduados (40%), seguido de graduandos (23.3%), mestrandos e pós-graduados (10%), mestres (6.7%) e finalmente doutorandos, doutores e com nível médio (3.3%). A Figura 6.1 apresenta os percentuais em termos de formação acadêmica e graduação máxima dos participantes.

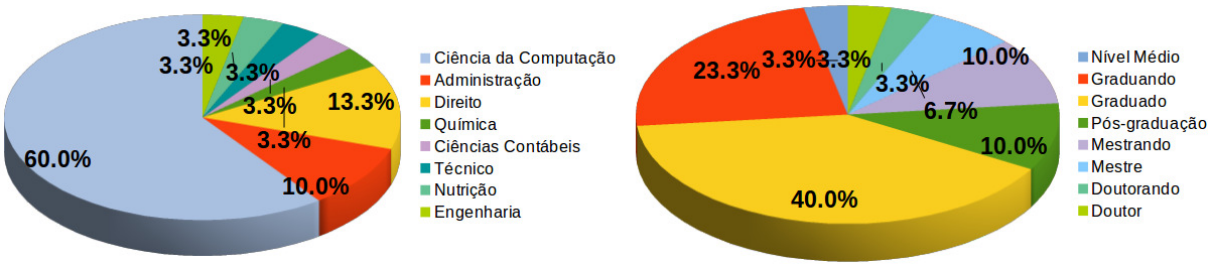

Figura 6.1: Perfil dos participantes: formação acadêmica e graduação máxima.

No que tange ao nível de experiência dos participantes, dentre os que possuem formação acadêmica em Ciências da Computação, 63.3% afirmaram ter algum tipo de experiência em desenvolvimento de sistemas, 33.3% afirmaram já ter trabalhado com alguma ferramenta MDE e 56.7% afirmaram conhecer a IDE Eclipse. Já dentre os que possuem formação em outras áreas acadêmicas, nenhum conhecia tais ferramentas.

6.6 Contexto e Instrumentação

O contexto do experimento é *in-vitro* (em laboratório), com participantes com perfis mistos entre estudantes e profissionais para resolver problemas do mundo real, mas com dados fictícios para especificação de restrições de mobilidade para dispositivos móveis.

A instrumentação do experimento foi composta de materiais disponibilizados para os participantes de forma impressa, como o roteiro para construção do objeto proposto no experimento e os mecanismos utilizados para registrar os resultados das medições. Por exemplo, para mensurar o tempo foram utilizados formulários que foram preenchidos pelos participantes a cada tarefa concluída.

Na operação das atividades do experimento, foram utilizados computadores localizados nas dependências do Laboratório de Sistemas Distribuídos Inteligentes (LSDi) e um notebook, todos consfigurados com sistema operacional Linux (Ubuntu e Elementary OS) de 64 bits.

6.7 Projeto do Experimento

O projeto do experimento demonstra como o experimento deve ser conduzido, além de determinar como os tratamentos são aplicados aos participantes utilizando os objetos [50].

O experimento realizado nesta avaliação considera dois fatores, quais são: esforço e percepção da qualidade. Cada fator foi analisado separadamente e os dados coletados em formulário. Os cenários utilizados como objeto de estudo foram aplicados aos participantes, para os quais apresentaram soluções, indicaram o tempo gasto em cada uma e, posteriormente, preencheram questionário de satisfação. Ao final, as soluções apresentadas pelos participantes para cada cenário foram avaliadas e qualificadas como:

- Corretas: cenários modelados sem nenhum erro;
- Incorretas: cenários modelados totalmente incorretos ou não modelados;

- Parcialmente Corretas: cenários modelados corretamente, mas apresentando alguns erros.

Previamente, o grupo de participantes recebeu treinamento sobre o domínio do problema (restrições de mobilidade) e sobre a ferramenta, simulando as atividades que deveriam executar durante o experimento, com conceitos básicos suficientes para especificação de restrições de mobilidade. Este treinamento teve duração de aproximadamente 15 minutos.

6.8 Operação do Experimento

Antes da execução do experimento foi realizada uma apresentação sobre o contexto que levou à sua realização, os principais conceitos envolvidos no domínio, a linguagem e a interface gráfica desenvolvidos, assim como uma visão geral de como esse experimento ia ser conduzido e esclarecimentos sobre questões relevantes para a realização das atividades, tomando cuidado para não dar muitos detalhes sobre a usabilidade da ferramenta e soluções dos cenários a serem resolvidos, o que comprometeria os fatores analisados, já que um conhecimento prévio da atividade a ser feita reduziria o tempo gasto e, consequentemente, influenciaria a satisfação do usuário para usabilidade da ferramenta.

O experimento consistiu em quatro etapas, sendo as três primeiras realizadas pelos participantes e a última pelo avaliador/pesquisador, todas realizadas através de um único formulário (contendo a identificação dos participantes, uma lista de cenários para solução e um questionário de satisfação) impresso e entregue aos participantes. A primeira etapa consistiu em preencher os dados de identificação de cada participante, para traçar seus perfis. Na segunda etapa, foram apresentados os três cenários objetos do estudo, para os quais os participantes deveriam apresentar uma solução de modelagem através da MobCons-AT, indicando o tempo gasto para resolução de cada um. A terceira etapa consistiu no preenchimento de um questionário para avaliar a eficiência e usabilidade da ferramenta. Para isso utilizou-se um questionário adaptado do *Post-Study System Usability Questionnaire* (PSSUQ), desenvolvido pela IBM para avaliação da satisfação do usuário e da usabilidade, sendo utilizado na indústria e academia [7, 31]. O PSSUQ utilizado possui 16 itens para

abordar características de usabilidade dos sistemas avaliados [45]. Foram consideradas as notas de avaliação de cada quesito apresentado **menor ou igual a três (3) como sendo positivo**, caso **igual a quatro (4) como sendo neutro** e **maior ou igual a cinco (5) como sendo negativo**. Finalmente, a quarta etapa, de responsabilidade do avaliador/pesquisador, consistiu em verificar os cenários modelados pelos usuários, indicando se foram modelados de forma correta, parcialmente correta ou incorreta.

O experimento foi executado em momentos distintos para que se pudesse alcançar o número de 30 participantes. Essa divisão ocorreu por questões de agenda dos participantes e por questões de limitação do ambiente para execução. Todos os experimentos foram executados conforme o projeto experimental descrito na Seção 6.7 e detalhado nesta seção.

6.9 Resultados

Os resultados do experimento estão estruturados em duas seções. A Seção 6.9.1 apresenta os resultados relacionados ao esforço obtido em cada etapa do experimento, enquanto a Seção 6.9.2 apresenta os resultados relacionados à percepção da qualidade. Os resultados são apresentados em forma de tabelas ou em forma de gráficos com a intenção de possibilitar uma melhor compreensão.

Na Seção Apêndices podem ser encontradas todas as planilhas e documentos elaborados no experimento contendo os dados compilados quanto ao perfil dos participantes (Apêndice F), avaliação de esforço (Apêndice G), avaliação da qualidade (Apêndice H) e questionário de avaliação fornecedido a todos os participantes do experimento (Apêndice I).

6.9.1 Resultados Relacionados ao Esforço

A Figura 6.2 mostra os resultados relacionados ao esforço para modelagem dos cenários apresentados por cada participante. O esforço foi consolidado em minutos, detalhando quanto tempo cada participante informou ter gasto para modelar os cenários níveis fácil, médio e difícil. Por exemplo, o participante 1 gastou cerca de 3 minutos e 39 segundos (3.65 minutos) para modelar o cenário nível fácil, 2 minutos

e 24 segundos (2.4 minutos) para o médio e 17 minutos e 4 segundos (17.07 minutos) para o difícil. Na análise dos tempos, observou-se que nem sempre o cenário médio levou mais tempo para ser modelado em comparação ao fácil, conforme informado pelos participantes 1 e 3, por exemplo. Isto pode se dar devido à experiência obtida na resolução do primeiro cenário, o que ajuda na resolução de outros exercícios. Na amostragem, aproximadamente 37% (11 participantes) conseguiram reduzir o esforço após resolução do primeiro cenário. Já o esforço para resolução do cenário difícil foi bastante grande, o que era esperado dado o nível de conceitos contidos no exemplo e o grau de exigência de conhecimento tanto por parte da MobCons-AT quanto do domínio e dos conceitos envolvidos. A Figura 6.3 mostra graficamente os resultados aferidos quanto ao esforço, por participante. Para cada tipo de cenário, é mostrado o esforço do participante, aferido em minutos.

Tempo de Modelagem dos Cenários Por Participante (hh:mm:ss)					Em Minutos		
Partipante	Fácil	Médio	Difícil		Fácil	Médio	Difícil
1	00:03:39	00:02:24	00:17:04		3.65	2.40	17.07
2	00:08:35	00:12:07	00:15:40		8.58	12.12	15.67
3	00:07:28	00:04:20	00:10:20		7.47	4.33	10.33
4	00:04:40	00:01:47	00:14:25		4.67	1.78	14.42
5	00:06:00	00:05:15	00:25:00		6.00	5.25	25.00
6	00:02:56	00:11:00	00:11:23		2.93	11.00	11.38
7	00:05:00	00:12:45	00:12:00		5.00	12.75	12.00
8	00:04:58	00:06:39	00:17:43		4.97	6.65	17.72
9	00:05:16	00:03:40	00:16:25		5.27	3.67	16.42
10	00:03:30	00:07:00	00:10:00		3.50	7.00	10.00
11	00:05:00	00:05:00	00:15:00		5.00	5.00	15.00
12	00:04:50	00:07:30	00:16:16		4.83	7.50	16.27
13	00:03:28	00:08:18	00:23:30		3.47	8.30	23.50
14	00:02:00	00:02:00	00:15:00		2.00	2.00	15.00
15	00:05:00	00:10:00	00:45:00		5.00	10.00	45.00
16	00:10:00	00:11:12	00:40:00		10.00	11.20	40.00
17	00:05:20	00:05:00	00:30:00		5.33	5.00	30.00
18	00:07:25	00:10:04	00:30:00		7.42	10.07	30.00
19	00:06:10	00:07:20	00:30:23		6.17	7.33	30.38
20	00:03:00	00:05:00	00:15:00		3.00	5.00	15.00
21	00:03:20	00:10:00	00:13:00		3.33	10.00	13.00
22	00:04:12	00:08:27	00:29:15		4.20	8.45	29.25
23	00:08:43	00:05:00	00:20:00		8.72	5.00	20.00
24	00:03:56	00:05:00	00:15:36		3.93	5.00	15.60
25	00:02:00	00:07:23	00:28:00		2.00	7.38	28.00
26	00:11:25	00:07:16	00:52:00		11.42	7.27	52.00
27	00:10:41	00:09:00	00:32:00		10.68	9.00	32.00
28	00:06:00	00:05:17	00:28:42		6.00	5.28	28.70
29	00:09:12	00:05:00	00:38:00		9.20	5.00	38.00
30	00:02:00	00:03:21	00:19:53		2.00	3.35	19.88

Figura 6.2: Resultados da avaliação relacionados ao esforço.

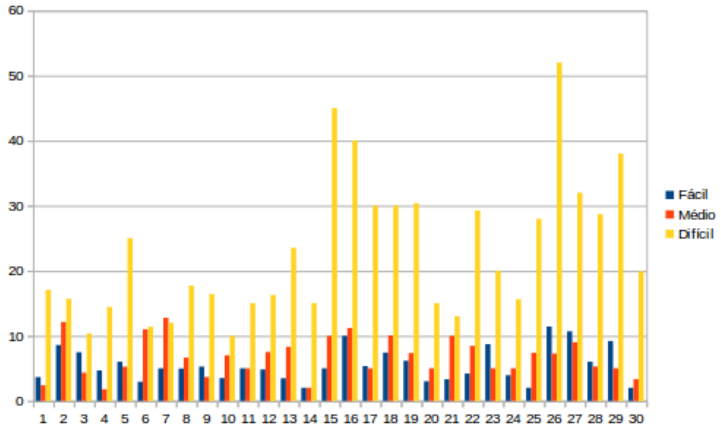

Figura 6.3: Resultados da avaliação relacionados ao esforço.

Com base em todos os tempos aferidos, a Figura 6.4 mostra os tempos médios obtidos pelos participantes para resolução de cada cenário (fácil, médio e difícil), em um comparativo aos tempos médios obtidos por especialistas para resolução dos mesmos cenários. Observa-se que o esforço médio de um usuário que tem contato pela primeira vez com a MobCons-AT é bem maior, evidentemente. Tal esforço aumenta de acordo com o grau de dificuldade exigido pelo cenário a ser modelado. O cenário fácil exigiu um esforço médio de 5.52 minutos, quando modelado pelos participantes do experimento, e apenas 1 minuto, quando modelado pelos especialistas. Já o cenário médio exigiu um esforço médio de 6.8 minutos pelos participantes e 1.5 minutos pelos especialistas. A dieferença se mostrou bem maior em relação ao cenário difícil, quando o esforço médio dos participantes foi 22.89 minutos, contra apenas 4 minutos médios dos especialistas.

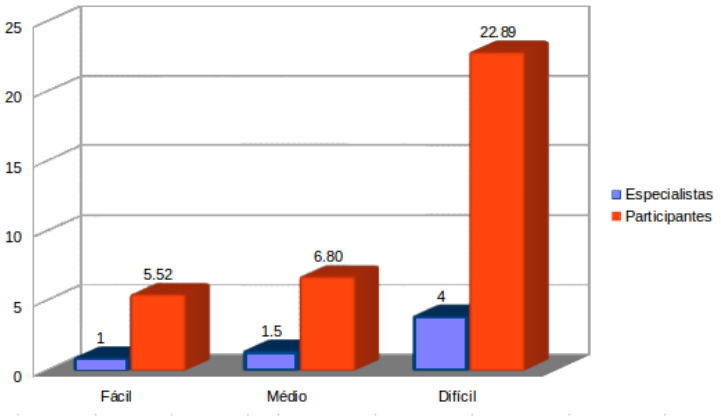

Figura 6.4: Tempos médios obtidos pelos participantes x especialistas.

Finalizando a análise com base nos tempos médios dos participantes, pode-se calcular um desvio padrão referente a cada tipo de cenário, para os quais obteve-se 1.17 (fácil), 0.67 (médio) e 1.99 (difícil), que representam medidas de dispersão em torno da média de esforço. Um baixo desvio padrão indica que os pontos dos dados tendem a estar próximos da média ou do valor esperado. Por outro lado, um alto desvio padrão indica que os pontos dos dados estão espalhados por uma ampla gama de valores. Os valores do desvio padrão obtidos indicam uma uniformidade dos participantes, ou seja, embora hajam alguns tempos bem distintos, a grande maioria aferida se manteve próximo da média do grupo. Em relação ao cenário fácil, em média, os tempos aferidos se afastaram 1.17 pontos, para mais ou para menos em relação à média de 5.52 minutos. Já em relação ao cenário médio, os tempos aferidos se afastaram 0.67 pontos em relação à média de 6.80 minutos. Por último, em relação ao cenário difícil, os tempos se afastaram 1.99 pontos em relação à media de 22.89 minutos. A Figura 6.5 mostra graficamente os tempos médios e o desvio padrão obtido para cada tipo de cenário.

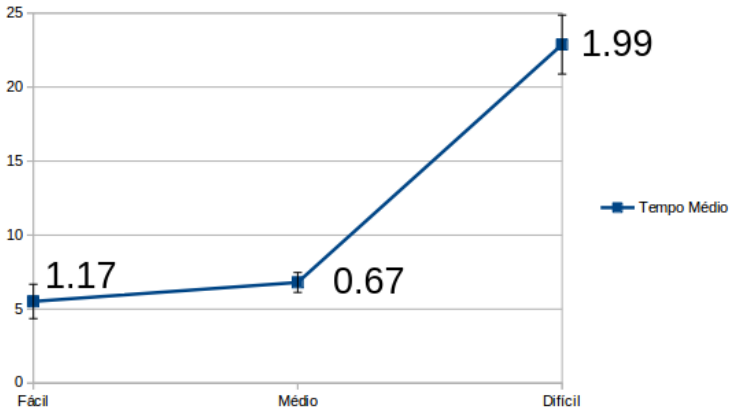

Figura 6.5: Desvio padrão dos tempos médios de esforço.

Complementando a avaliação do esforço dos participantes, as Figuras 6.6 e 6.7 mostram as taxas de acertos das modelagens feitas por cada participante, para cada cenário proposto. São mostrados detalhadamente cada resultado por participante, além de uma consolidação desses resultados, por tipo de cenário (fácil, médio e difícil), dando suas frequências absolutas (FA) e relativas (FR). Pode ser verificado que o cenário fácil foi modelado corretamente por 25 participantes (83.3%), não teve nenhuma modelagem incorreta e foi modelado parcialmente correto por 5 participantes (16.7%). O cenário médio, por sua vez, foi modelado corretamente

por 17 participantes (56.7%), incorretamente por 2 participantes (6.7%) e parcialmente correto por 11 participantes (36.7%). Já o cenário difícil foi modelado corretamente por apenas 3 participantes (10%), dentre os quais são um mestrando, um doutor e um que já possuía amplo conhecimento no domínio e da ferramenta, incorretamente por 6 participantes (20%) e parcialmente correto pela grande maioria, ou seja, 21 participantes (70%). Isto se deveu ao fato da maior riqueza de conceitos envolvidos no cenário, o que levou os participantes a acertarem os conceitos mais óbvios, errando os que exigiam um grau maior de conhecimento.

Modelagens Corretas Por Participante			
Participante	**Fácil**	**Médio**	**Difícil**
1	Sim	Sim	Parcialmente
2	Sim	Sim	Não
3	Sim	Sim	Parcialmente
4	Sim	Sim	Sim
5	Sim	Sim	Parcialmente
6	Sim	Parcialmente	Não
7	Sim	Sim	Não
8	Sim	Sim	Parcialmente
9	Sim	Sim	Parcialmente
10	Sim	Sim	Parcialmente
11	Sim	Sim	Sim
12	Sim	Sim	Parcialmente
13	Sim	Sim	Parcialmente
14	Sim	Parcialmente	Parcialmente
15	Sim	Sim	Parcialmente
16	Sim	Não	Não
17	Sim	Sim	Parcialmente
18	Sim	Parcialmente	Parcialmente
19	Sim	Parcialmente	Parcialmente
20	Sim	Sim	Parcialmente
21	Sim	Sim	Sim
22	Parcialmente	Parcialmente	Não
23	Sim	Parcialmente	Não
24	Parcialmente	Parcialmente	Parcialmente
25	Parcialmente	Não	Parcialmente
26	Parcialmente	Parcialmente	Parcialmente
27	Sim	Sim	Parcialmente
28	Parcialmente	Parcialmente	Parcialmente
29	Sim	Parcialmente	Parcialmente
30	Sim	Parcialmente	Parcialmente

Fácil	**FA**	**FR**
Sim	25	83.3%
Não	0	0.0%
Parcialmente	5	16.7%
TOTAL	30	100%

Médio	**FA**	**FR**
Sim	17	56.7%
Não	2	6.7%
Parcialmente	11	36.7%
TOTAL	30	100%

Difícil	**FA**	**FR**
Sim	3	10.0%
Não	6	20.0%
Parcialmente	21	70.0%
TOTAL	30	100%

Figura 6.6: Taxas de acertos das modelagens feitas por cada participante.

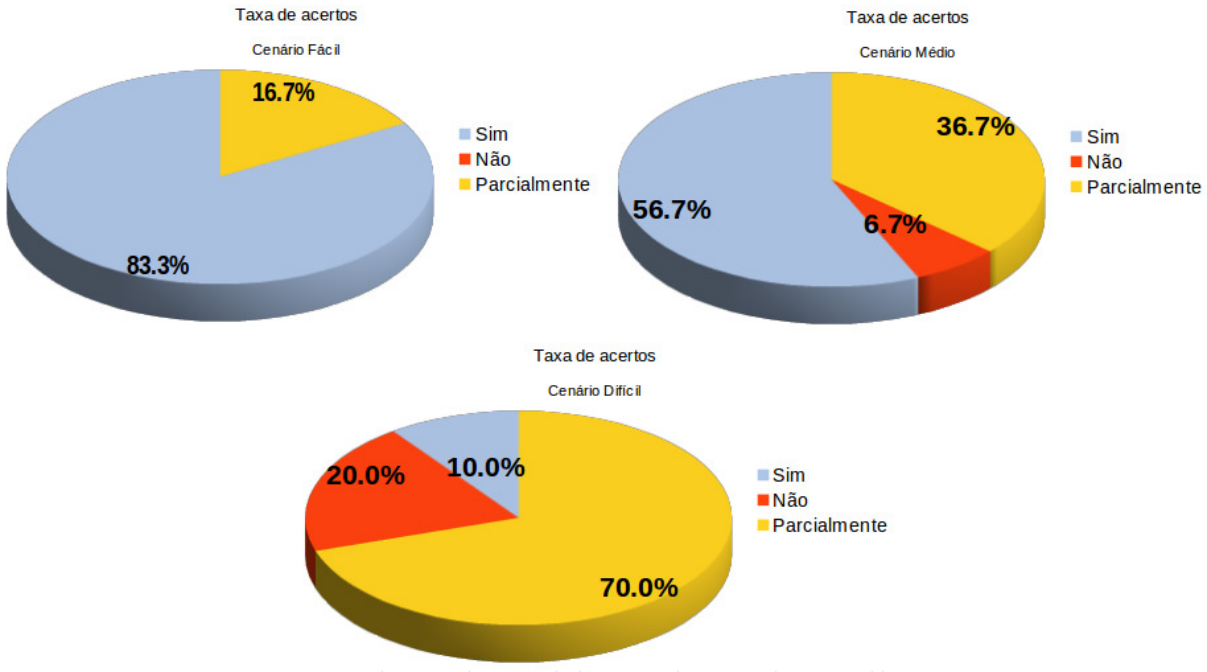

Figura 6.7: Taxas de acertos das modelagens feitas por cada participante.

6.9.2 Resultados Relacionados à Percepção da Qualidade

A percepção da qualidade, por ser algo subjetivo, foi aferida através da aplicação aos participantes do questionário PSSUQ, que pode ser encontrado no Apêndice I. O PSSUQ produz 4 grupos de pontuação: um grupo geral (OVERALL), obtido através dos dezesseis itens do questionário, e três grupos obtidos por sub-escalas. A primeira é a utilidade do sistema (SYSUSE), obtida através dos itens de 1 a 6 do questionário, a segunda é a qualidade da informação (INFOQUAL), obtida através dos itens de 7 a 12 do questionário e a terceira é a qualidade da interface (INTERQUAL), obtida através dos itens 13 a 16 do questionário. Cada item é medido em uma escala Likert [33] de 7 pontos, cujas respostas variam conforme abaixo:

1. Concordo Totalmente
2. Concordo Parcialmente
3. Concordo
4. Neutro
5. Discordo

6. Discordo Parcialmente

7. Discordo Totalmente

Embora utilize números em sua composição, nas escalas Likert eles são meramente ordenações de posição, não tendo nenhum sentido de medida numérica (quantitativa) propriamente dito. Em outras palavras, uma resposta com valor 4 não apresenta maior valor ou peso do que uma resposta marcada com 1. Por esse motivo, não cabe a utilização de médias ou desvio padrão em suas análises. No questionário, adotou-se que pontuações mais baixas indicam uma classificação geral positiva.

Os resultados obtidos no questionário foram analisados através da frequência absoluta de cada item, por questão, informação esta que nos diz quantos participantes deram nota 1, nota 2, nota 3 e assim sucessivamente. A Figura 6.8 mostra os resultados compilados. Observa-se que em todos os grupos de pontuação (OVERALL, SYSUSE, INFOQUAL e INTERQUAL), a frequência relativa é maior dentre os itens positivos (de 1 a 3), o que indica maior satisfação dos usuários quanto a usabilidade da ferramenta. Embora hajam pontos a melhorar na MobCons-AT, os itens negativos (de 5 a 7) não tiveram grande representatividade na amostragem, apresentando baixas frequências relativas.

OVERALL		
	FA	FR
1	216	45.0%
2	159	33.1%
3	60	12.5%
4	38	7.9%
5	3	0.6%
6	1	0.2%
7	3	0.6%
TOTAL	480	100%

SYSUSE		
	FA	FR
1	87	48.3%
2	63	35.0%
3	23	12.8%
4	6	3.3%
5	1	0.6%
6	0	0.0%
7	0	0.0%
TOTAL	180	100%

INFOQUAL		
	FA	FR
1	65	36.1%
2	60	33.3%
3	27	15.0%
4	24	13.3%
5	1	0.6%
6	1	0.6%
7	2	1.1%
TOTAL	180	100%

INTERQUAL		
	FA	FR
1	64	53.3%
2	36	30.0%
3	10	8.3%
4	8	6.7%
5	1	0.8%
6	0	0.0%
7	1	0.8%
TOTAL	120	100%

Figura 6.8: Resultados da avaliação relacionados à percepção da qualidade.

A Figura 6.9 mostra os resultados obtidos graficamente, desta vez agrupando os itens como positivos (de 1 a 3), neutro (4) e negativos (de 5 a 7). Por exemplo, o indicador OVERALL teve 90% de avaliações positivas, o indicador SYSUSE apresentou 96%, o indicador INFOQUAL apresentou 84% e o INERQUAL apresentou 91%.

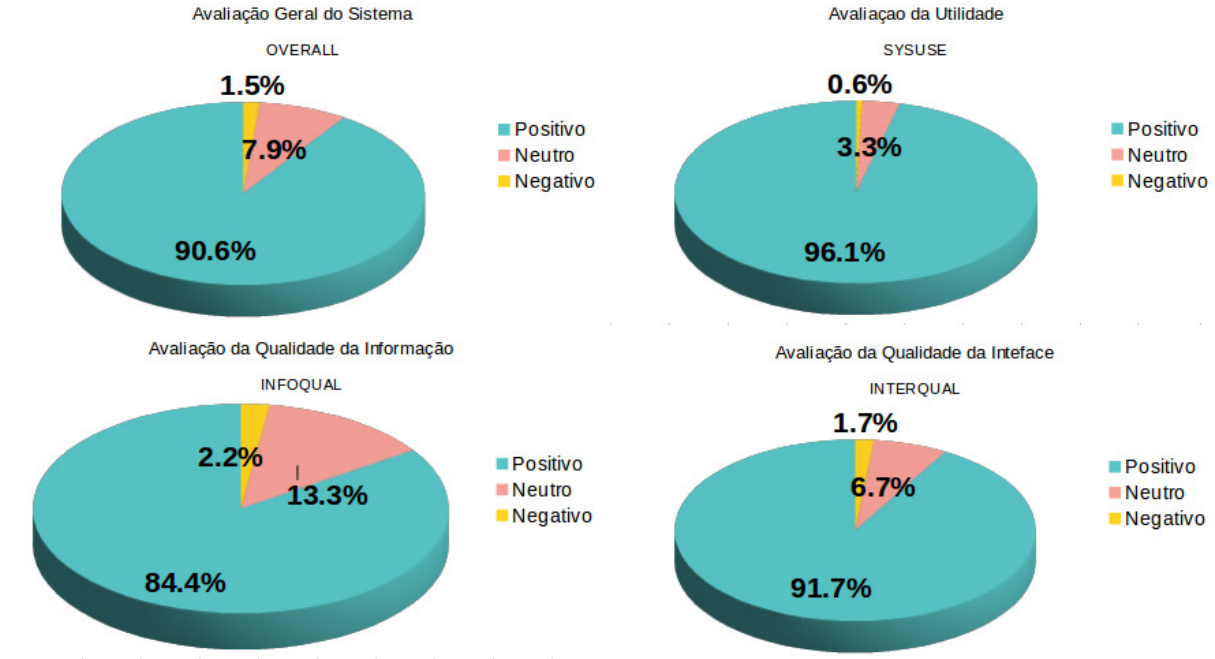

Figura 6.9: Resultados da avaliação relacionados à percepção da qualidade.

Todos os dados coletados corroboram o fato de que, no geral, a MobCons-AT conseguiu atender às expectativas de boa usabilidade e intuitividade, se mostrando uma ferramenta útil na modelagem de cenários que envolvam a restrição de mobilidade de dispositivos móveis.

6.10 Validade

Uma questão importante sobre os resultados obtidos em um experimento é sua confiabilidade, ou seja, até que ponto eles podem ser considerados válidos [50, 54]. Aspectos relacionados à validade devem ser previstos durante a fase de planejamento de um experimento para tentar evitar situações que possam invalidá-lo [54]. As validações comumente utilizadas na literatura são: de conclusão, interna, de construção e externa [50].

6.10.1 Validação de Conclusão

A validação de conclusão procura garantir que conclusões corretas sobre os resultados possam ser alcançadas. Alguns dos problemas mais comuns recorrentes às conclusões que foram consideradas neste experimento foram apresentados juntamente com a forma de mitigá-los:

- Confiabilidade das medições: Seu objetivo é garantir a confiabilidade do processo de medição, sendo ideal que a medição não utilize julgamento humano. Por isso, medições objetivas que não necessitam de interpretação humana são mais confiáveis. Neste trabalho, utilizou-se uma medida objetiva (esforço) e outra medida subjetiva (percepção da qualidade). Este problema pode ter afetado o experimento na medida que os próprios participantes aferiram seus esforços nas atividades de modelagem, embora estivessem sob supervisão do avaliador. Talvez o ideal seria prover alguma forma de aferição automática que não dependesse dos participantes;

- Confiabilidade da aplicação dos tratamentos: Deve garantir que todos os participantes utilizem os mesmos tratamentos. Neste trabalho, utilizou-se um mesmo tratamento para todos os participantes, descartando-se a possibilidade de aplicações diferentes, pois todos os participantes foram submetidos à mesma abordagem;

- Distúrbio na configuração do experimento: Elementos externos ao estudo podem interferir nos resultados. Para mitigar essa ameaça, o experimento foi executado em ambientes tranquilos e com acesso controlado para favorecer a concentração dos participantes.

6.10.2 Validação Interna

A validação interna procura garantir que, durante o experimento, só haja interferência controlada entre o tratamento aplicado e os resultados alcançados, ou seja, se os resultados foram gerados pelos tratamentos aplicados sem interferência controlada. Uma lista de ameaças à validade interna, assim como tratá-las serão apresentadas:

- História: Acontecimentos históricos podem interferir nos resultados, pois podem influenciar diretamente os participantes. Por exemplo, indisposição depois de um final de semana e/ou euforia depois de grandes eventos. Para tentar mitigar esses problemas, as datas para a realização do experimento foram marcadas antecipadamente, conforme disponibilidade dos participantes, tentando-se evitar feriados, finais de semanas e dias com eventos culturais e sociais que pudessem afetar os participantes do experimento;

- Mortalidade: Diz respeito à quantidade de participantes que por desistência não concluem o experimento. No caso deste trabalho, não houve nenhuma desistência de participante;

- Maturação: Essa ameaça está relacionada ao comportamento ou reação dos participantes com o passar do tempo. Tratam de alterações naturais (em vez de impostas pelo experimentador) que ocorrem como resultado da passagem normal do tempo. Por exemplo, quanto mais tempo passar em um estudo, mais provável é de os sujeitos se tornarem cansados e aborrecidos, mais ou menos motivados em função da fome ou da sede, mais velhos, etc. Para mitigar esses problema, o experimento foi planejado de modo a não ser demorado em demasia;

- Ameaças Sociais: Permite garantir que os resultados não sofram interferência de fatores sociais como rivalidade, amizade ou inimizade. Para evitar esses problemas, foi incluído um breve resumo sobre o objetivo do experimento na ficha de identificação do perfil do participante. Esse resumo ressaltava que o experimento não se tratava de competição ou de uma avaliação pessoal do participante. Um reforço sobre essas questões, em forma de uma apresentação, foi realizado depois que os participantes foram selecionados;

6.10.3 Validação de Construção

A validade de construção considera os relacionamentos entre a teoria e a observação, ou seja, consiste em garantir que o tratamento reflita a causa de maneira satisfatória, assim como que o resultado reflita o efeito de forma satisfatória. Esta validação está ligada diretamente aos aspectos relevantes do projeto do experimento do ponto de vista dos pesquisadores e dos participantes. Ameaças como a expectativa

do pesquisador, má definição da base teórica ou da definição do processo de experimentação e as possíveis suposições dos participantes, quanto aos objetivos, devem ser consideradas. Para mitigá-las, os pesquisadores evitaram auxiliar os participantes durante o experimento e os participantes foram instruídos a seguir o experimento de maneira imparcial para que sua opinião pessoal não interferisse nos resultado.

6.10.4 Validação Externa

A validade externa trata da generalização dos resultados do experimento. Alguns problemas e como foram tratados são apresentados a seguir:

- Seleção dos participantes: Pode ser um problema, pois a amostra de participantes selecionada pode não ser representativa frente à população desejada. Para tentar ampliar a capacidade de generalização dos resultados, foram utilizados participantes de ensino médio, de graduação, pós-graduação e alguns profissionais, não só da área acadêmica de Computação, como de outras áreas;

- Ferramentas inadequadas: O problema ocorre na escolha de ferramentas que não sejam adequadas e de alguma maneira favoreçam alguns dos tratamentos executados. As ferramentas selecionadas são amplamente conhecidas e utilizadas tanto para atividades acadêmicas como profissionais. Entretanto, os participantes das áreas que não a Computação não conheciam as ferramentas e isso, obviamente, causou um certo impacto na avaliação, principalmente no início do experimento, quando os participantes tinham os primeiros contatos com as ferramentas;

- História: Esse problema está relacionado com os dias de execução do experimento. Fatores externos podem afetar os resultados, como por exemplo, se for escolhido algum dia que, por algum evento único (manifestação que atrapalhe a chegada dos participantes), leve os participantes a saírem da sua rotina. Essa mudança de rotina pode influenciar os resultados. Para diminuir esse risco, as datas do experimento foram escolhidas e combinadas com cada participante de forma cautelosa a evitar situações atípicas;

A ordem de prioridade das validações é um aspecto importante que deve ser considerado, sendo determinada em função do objetivo do experimento. Para experimentos aplicados, alvo da maioria dos experimentos da área de Engenharia de Software e, por consguinte, alvo deste estudo, a ordem de prioridade das validações deve ser: interna, externa, de construção e de conclusão [40]. Apesar da validade externa ter sido prejudicada devido às ferramentas utilizadas no experimento não serem do conhecimento dos participantes das áreas acadêmicas diferentes de Computação e da validade de conclusão também ter sido afetada pela aferição dos tempos sob resposabilidade dos participantes, as demais validações foram bem tratadas.

6.11 Análise dos Resultados e Discussões

Os resultados apresentados na Seção 6.9 foram analisados e interpretados por meio de observação. A análise e interpretação dos resultados serão distribuídas em seções, assim como ocorreu na apresentação dos resultados. A primeira Seção 6.11.1 apresentará a análise sobre os resultados relacionados ao esforço e a Seção 6.11.2 apresentará a análise sobre a percepção de qualidade.

6.11.1 Análise dos Resultados Relacionados ao Esforço

Analisando os dados compilados na Seção 6.9.1, observou-se que a média de esforço dos participantes ficou um pouco distante a média dos especialistas, o que era esperado dado ser o primeiro contato dos participantes com a MobCons-AT e com os conceitos do domínio. Após a resolução do primeiro cenário, 37% dos participantes conseguiram reduzir o esforço para modelar o cenário de deficuldade média. Embora os tempos para resolução dos cenários tenham sido um pouco altos, comparados aos de especialistas, o desvio padrão foi baixo, indicando uniformidade entre os participantes, ou seja, mesmo os de formação acadêmica diferentes de Computação conseguiram manter uma proximidade da média dos demais.

Finalmente, verificou-se que a taxa de acerto de modelagem dos cenários fácil e médio pelos participantes ficou acima de 50%, indicando boa produtividade. Já o cenário difícil, por exigir maior conhecimento da MobCons-AT e dos conceitos

de domínio, só foi modelado corretamente por participantes da área de Computação, com nível de graduação maiores (mestre e doutor) ou com amplo conhecimento do domínio.

6.11.2 Análise dos Resultados Relacionados à Percepção da Qualidade

Analisando os dados compilados na Seção 6.9.2, observou-se que a MobCons-AT teve boa aceitação em todos os quesitos avaliados referentes à percepção da qualidade, mostrando boa usabilidade, qualidade de informação e de interface. Com exceção do indicador de qualidade da informação, que teve 84% de avaliações positivas, os demais indicadores obtiveram aceitação de mais de 90% por parte dos participantes.

Alguns participantes também foram questionados a respeito da utilização da ferramenta. Considerando que a ferramenta é um protótipo e apresenta algumas limitações quanto à usabilidade, que foram apontadas pelos participantes, a grande maioria afirmou que utilizaria a MobCons-AT em trabalhos comerciais. De maneira geral, o protótipo da MobCons-AT se demonstrou aceitável, mas para que possa ser considerado comercialmente viável, deve apresentar melhorias de usabilidade e validação.

6.11.3 Discussões

Nesta seção serão discutidos pontos que foram levantados pelos participantes durante a execução do experimento. Destacam-se muitos pontos a melhorar, muitos elogios e alguns erros que foram citados ao longo da operação. Os principais pontos citados foram:

- Para ligar uma regra a uma restrição, é preciso clicar primeiramente na regra e depois na restrição. Esta propriedade foi questionada por um participante que acharia melhor ter tanto essa opção quanto o contrário (clicar na restrição e depois na regra). Fora isso, o participante achou a MobCons-AT muito boa, com excelente funcionalidade e interface, facilidade de uso e intuitividade;

- Ao adicionar os objetos no diagrama de modelagem, a MobCons-AT exibe um nome atribuído automaticamente ao objeto (por exemplo MU1, Group1, MUType2) e não o nome especificado pelo participante (viaturas, grupo de veículos, etc). Este fato foi citado diversas vezes pelos participantes e requer verificação para melhoria;

- Alguns participantes que tiveram a primeira experiência com ferramenta MDE informaram ter tido facilidade no aprendizado e ficaram surpresos com a rapidez para elaborar um cenário;

- Dois participantes, uma da área de Administração e outro do Direito, questionaram se a MobCons-AT não poderia ter sido desenvolvida em português, já que não dominam o inglês. Segundo eles, isso dificultou um pouco a usabilidade;

- Alguns participantes reclamaram do tamanho da tela disponível para modelar os cenários. Segundo eles, a área era muito pequena e dificultava a organização dos objetos;

- Os participantes acharam os ícones dos objetos bem representativos, de fácil compreensão, o que ajudava bastante na intuitividade;

- As formas como são feitas as ligações (relacionamentos) entre os objetos são simples e de fácil interpretação;

- Quase todos os participantes tiveram dificuldade com o uso de alguns objetos presentes nos cenários de estudo, tais como os contextos de Grupo, MUType, Coordenadas e as restrições de Acesso e Distância Mínima;

- Alguns participantes sentiram dificuldade com o uso da tela de propriedades dos objetos, onde são inseridos os valores de velocidade, distâncias, etc. Houve uma sugestão para que esses campos pudessem ser mostrados automaticamente assim que os objetos fossem inseridos no diagrama;

- Muitos participantes acharam que o experimento exigia muito conhecimento e afirmaram ser muita informação para pouco tempo de operação. Os participantes com experiência em desenvolvimento e que conheciam modelagem de dados tinham mais facilidade em absorver os conceitos;

- As mensagens de erro poderiam estar melhor visíveis, ser mais claras e destacar o objeto com erro de modelagem;

- Como todos os contextos possuem um campo "descrição", dois participantes mais experientes acharam pouco intuitivo entender que no contexto MUType o campo "descrição"é onde deve ser inserido o tipo de dispositivo móvel que define o contexto. Deveria ter sido criado um atributo específico para esse fim;

- Um dos participantes, da área de Computação e com alto grau de formação, elogiou o fato da MobCons-AT não permitir inserir nem relacionar objetos em lugares semanticamente incorretos. Ponto bastante positivo e destacado;

- O único participante com doutorado, inclusive na área de MDE, sugeriu evitar abreviações no nome de metaclasses (MU) e realizar o experimento com mais cenários. Fora isso, achou que a interface está bem organizada e intuitiva e comentou que a adaptação do usuário vai melhorando com o manuseio da ferramenta, tornando mais agradável seu uso com o tempo;

- De uma forma geral, os participantes comentaram que a modelagem gráfica torna fácil e rápida a definição de regras, além do escopo de objetos implementados ser bem amplo.

Observa-se nos comentários dos participantes que, de fato, existem pontos a melhorar na MobCons-AT. Algumas sugestões são bem pertinentes e merecem atenção e correções. Outras apenas refletem gostos pessoais e não necessariamente agregariam valor à ferramenta se implementadas. Além das sugestões de melhoria e críticas, a MobCons-AT recebeu elogios, até mesmo do participante com mais alta graduação (doutorado em MDE), o que para os pesquisadores teve um peso maior.

7 Conclusões e Trabalhos Futuros

O avanço tecnológico na área da computação móvel e pervasiva abre oportunidade para a análise em tempo real de dados de localizaçao, velocidade, temperatura, etc., informações muito úteis para o entendimento da rotina da vida nas grandes cidades, permitindo que tanto o setor público quanto o privado possam detectar possíveis irregularidades e redistribuir recursos.

Embora existam alguns trabalhos desenvolvidos nessa temática, ainda é utilizada a abordagem tradicional da engenharia de software, onde uma equipe de desenvolvedores é necessária em todas as fases do projeto, desde a análise do problema até a implementação de melhorias e alterações pós entrega. Há, portanto, uma lacuna referente a uma solução que possa ser modelada em alto nível pelo usuário final, reduzindo custos e tempo.

Este trabalho apresenta a MobCons-AT, uma ferramenta de autoria que, através de uma DSML chamada MobCons-SL, permite a especificação de restrições de mobilidade para dispositivos móveis em alto nível. Através de estudos de caso em cinco cenários do mundo real, observa-se que a linguagem proposta é capaz de modelar muitas situações que envolvam esse domínio. Além disso, a interface gráfica, embora ainda um protótipo, consegue ilustrar as relações entre todos os conceitos envolvidos no metamodelo proposto, cabendo ao usuário final somente aprender cada representação visual e projetar os modelos.

As principais contribuições deste trabalho são:

- A investigação do estado da arte em ferramentas de autoria que implementem uma DSML para o apoio à especificação de restrições de dispositivos móveis em alto nível por parte do usuário final. Através desta investigação, ficou evidenciada a escassez de mecanismos específicos para esse fim, o que motivou o desenvolvimento da MobCons-AT;
- O desenvolvimento da linguagem MobCons-SL, que abstrai a complexidade de especificar restrições de mobilidade de dispotivos móveis, permitindo aos

usuários modelar cenários sem que haja erros de sintaxe ou de digitação no artefato gerado;

- O desenvolvimento de uma ferramenta de autoria com interface gráfica que, através do uso da linguagem MobCons-SL desenvolvida, permite a modelagem de cenários em alto nível nos quais seja requerida a restrição de mobilidade de dispositivos móveis.

Através da apresentação de cinco estudos de casos para uso da MobCons-AT, observou-se que ela é capaz de modelar vários cenários e que algumas modelagens mais complexas exigem um bom conhecimento, tanto do domínio de restrições de mobilidade e dos conceitos envolvidos, quanto da ferramenta proposta.

Apesar dos diversos objetivos alcançados durante o desenvolvimento deste trabalho, há algumas possibilidades de melhorias que foram observadas. Os trabalhos futuros que podem contribuir para a evolução desta pesquisa são:

- Implementação de novas classes `Context`, `Restriction` e `TemporalCondition` para tornar a linguagem e a ferramenta cada vez mais completas e abrangerem um número maior de cenários e situações;
- Integração da ferramenta com mapas para inserção de coordenadas geográficas reais ao usar o contexto `Area`, bem como as restrições `Path` e `Punctuality`. Atualmente é necessário o acesso a uma aplicação externa que forneça esse dados do mundo real para que sejam, então, inseridos nos modelos produzidos;
- Alteração do ferramenta para permitir a reutilização de um mesmo objeto `TemporalCondition`, quando associado a diversas regras, necessidade detectada através dos estudos de caso e discutida na seção 5.3.2;
- Modelagem para permitir a detecção de padrões de mobilidade (concentração, dispersão, comboio, etc.).

Referências Bibliográficas

[1] Atl: Atlas transformation language documentation. https://www.eclipse.org/atl/documentation/.

[2] M. A. Al-Khedher. Hybrid gps-gsm localization of automobile tracking system. *arXiv preprint arXiv:1201.2630*, 2012.

[3] M. A. Al-Taee, O. B. Khader, and N. A. Al-Saber. Remote monitoring of vehicle diagnostics and location using a smart box with global positioning system and general packet radio service. In *2007 IEEE/ACS International Conference on Computer Systems and Applications*, pages 385–388, May 2007.

[4] I. M. Almomani, N. Y. Alkhalil, E. M. Ahmad, and R. M. Jodeh. Ubiquitous gps vehicle tracking and management system. In *2011 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT)*, pages 1–6, Dec 2011.

[5] A. Antoniou, A. Georgiou, P. Kolios, C. Panayiotou, and G. Ellinas. An event-based bus monitoring system. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 2882–2887, Oct 2014.

[6] B. F. Arndt et al. Mme-mdd: um método para manutenção e evolução de sistemas baseados no mdd. 2016.

[7] A. Bangor, P. T. Kortum, and J. T. Miller. An empirical evaluation of the system usability scale. *Intl. Journal of Human–Computer Interaction*, 24(6):574–594, 2008.

[8] M. Behzad, A. Sana, M. Khan, Z. Walayat, U. Qasim, Z. Khan, and N. Javaid. Design and development of a low cost ubiquitous tracking system. *Procedia Computer Science*, 34(Supplement C):220 – 227, 2014.

[9] V. A. Bollati, J. M. Vara, Á. Jiménez, and E. Marcos. Applying mde to the (semi-) automatic development of model transformations. *Information and Software Technology*, 55(4):699–718, 2013.

[10] M. Brambilla, J. Cabot, and M. Wimmer. Model-driven software engineering in practice. *Synthesis Lectures on Software Engineering*, 3(1):1–207, 2017.

[11] J.-M. Bruel, B. Combemale, I. Ober, and H. Raynal. Mde in practice for computational science. *Procedia Computer Science*, 51:660–669, 2015.

[12] J. S. Cuadrado, J. L. C. Izquierdo, and J. G. Molina. Applying model-driven engineering in small software enterprises. *Science of Computer Programming*, 89:176–198, 2014.

[13] G. Cugola and A. Margara. Processing Flows of Information: From Data Stream to Complex Event Processing. *ACM Computing Surveys*, 44(3):1–62, jun 2012.

[14] K. Czarnecki and S. Helsen. Classification of model transformation approaches. In *Proceedings of the 2nd OOPSLA Workshop on Generative Techniques in the Context of the Model Driven Architecture*, volume 45, pages 1–17. USA, 2003.

[15] A. R. Da Silva. Model-driven engineering: A survey supported by the unified conceptual model. *Computer Languages, Systems & Structures*, 43:139–155, 2015.

[16] O. Etzion. Event processing: past, present and future. *Proceedings of the VLDB Endowment*, 3(1-2):1651–1652, 2010.

[17] I. Flouris, N. Giatrakos, A. Deligiannakis, M. Garofalakis, M. Kamp, and M. Mock. Issues in complex event processing: Status and prospects in the big data era. *Journal of Systems and Software*, 127:217–236, 2017.

[18] M. Fowler. *Domain-specific languages*. Pearson Education, 2010.

[19] J. V. Guinelli, A. de Souza Rosa, C. E. Pantoja, R. Choren, N. Friburgo-RJ-Brasil, and R. de Janeiro-RJ-Brasil. Uma metodologia para apoio ao projeto de banco de dados geográficos utilizando a mda. *X Simpósio Brasileiro de Sistemas de Informação*, 2014.

[20] D. Harel and B. Rumpe. Modeling languages: Syntax, semantics and all that stu. Technical report, Technical report, 2000.

[21] C. M. Hartmann. Linguagem e ferramenta de autoria para promover o desenvolvimento de perícias em xadrez. 2005.

[22] J. Heering and M. Mernik. Domain-specific languages in perspective. *Software Engineering [SEN]*, (E0702), 2007.

[23] F. Hermans, M. Pinzger, and A. Van Deursen. Domain-specific languages in practice: A user study on the success factors. In *International Conference on Model Driven Engineering Languages and Systems*, pages 423–437. Springer, 2009.

[24] S. P. Joy, V. S. Sunitha, V. R. S. Devi, A. Sneha, S. Deepak, and A. J. Raju. A novel security enabled speed monitoring system for two wheelers using wireless technology. In *2016 International Conference on Circuit, Power and Computing Technologies (ICCPCT)*, pages 1–7, March 2016.

[25] M. P. R. Junior. *DG2CEP: An On-line Algorithm for Real-time Detection of Spatial Clusters from Large Data Streams through Complex Event Processing*. PhD thesis, PUC-Rio, 2017.

[26] B. Kim, S. Lee, Y. Lee, I. Hwang, Y. Rhee, and J. Song. Mobiiscape: Middleware support for scalable mobility pattern monitoring of moving objects in a large-scale city. *Journal of Systems and Software*, 84(11):1852–1870, 2011.

[27] A. G. Kleppe, J. Warmer, J. B. Warmer, and W. Bast. *MDA explained: the model driven architecture: practice and promise*. Addison-Wesley Professional, 2003.

[28] N. Koch. Transformation techniques in the model-driven development process of uwe. In *Workshop proceedings of the sixth international conference on Web engineering*, page 3. ACM, 2006.

[29] E. J. Lee and K. H. Ryu. Design of vehicle information management system for effective retrieving of vehicle location. In *International Conference on Computational Science and Its Applications*, pages 998–1007. Springer, 2005.

[30] P.-R. Lei. A framework for anomaly detection in maritime trajectory behavior. *Knowledge and Information Systems*, 47(1):189–214, 2016.

[31] J. R. Lewis. Ibm computer usability satisfaction questionnaires: psychometric evaluation and instructions for use. *International Journal of Human-Computer Interaction*, 7(1):57–78, 1995.

[32] J. R. Lewis. Psychometric evaluation of the pssuq using data from five years of usability studies. *International Journal of Human-Computer Interaction*, 14(3-4):463–488, 2002.

[33] R. Likert. A technique for the measurement of attitudes. *Archives of psychology*, 1932.

[34] D. Luckham, R. Schulte, J. Adkins, P. Bizarro, A. Mavashev, and P. Niblett. Event processing glossary version 2.0 (2011). *URL http://www. complexevents. com/wp-content/uploads/2011/08/EPTS_Event_Processing_Glossary_v2. pdf*.

[35] J. Luoma, S. Kelly, and J.-P. Tolvanen. Defining domain-specific modeling languages: Collected experiences. In *4 th Workshop on Domain-Specific Modeling*, 2004.

[36] S. J. Mellor. *MDA distilled: principles of model-driven architecture*. Addison-Wesley Professional, 2004.

[37] M. Mernik, J. Heering, and A. M. Sloane. When and how to develop domain-specific languages. *ACM computing surveys (CSUR)*, 37(4):316–344, 2005.

[38] T. Murray. Authoring intelligent tutoring systems: An analysis of the state of the art. *International Journal of Artificial Intelligence in Education (IJAIED)*, 10:98–129, 1999.

[39] T. Murray, S. Blessing, and S. Ainsworth. *Authoring tools for advanced technology learning environments: Toward cost-effective adaptive, interactive and intelligent educational software*. Springer Science & Business Media, 2003.

[40] P. A. M. d. OLIVEIRA. Athena: uma arquitetura e uma ferramenta para auxiliar o desenvolvimento de sistemas baseados em inteligência computacional. 2016.

[41] R. R. Oliveira, F. C. Noguez, C. A. Costa, J. L. Barbosa, and M. P. Prado. Swtrack: An intelligent model for cargo tracking based on off-the-shelf mobile devices. *Expert Systems with Applications*, 40(6):2023 – 2031, 2013.

[42] F. S. Parreiras. *Semantic Web and model-driven engineering*. John Wiley & Sons, 2012.

[43] D. Punetha and V. Mehta. Protection of the child/ elderly/ disabled/ pet by smart and intelligent gsm and gps based automatic tracking and alert system. In *2014*

International Conference on Advances in Computing, Communications and Informatics (ICACCI), pages 2349–2354, Sept 2014.

[44] L. Salmon and C. Ray. Design principles of a stream-based framework for mobility analysis. *GeoInformatica*, 2016.

[45] J. Sauro and J. R. Lewis. *Quantifying the user experience: Practical statistics for user research*. Morgan Kaufmann, 2016.

[46] S. Shahrivari. Beyond Batch Processing: Towards Real-Time and Streaming Big Data. *CoRR*, abs/1403.3, 2014.

[47] M. Sharples, J. Du Boulay, B. Teather, D. Teather, N. Jeffery, and G. Du Boulay. The mr tutor: Computer-based training and professional practice. *COGNITIVE SCIENCE RESEARCH PAPER-UNIVERSITY OF SUSSEX CSRP*, 1995.

[48] B. H. Soleimani, E. N. De Souza, C. Hilliard, and S. Matwin. Anomaly detection in maritime data based on geometrical analysis of trajectories. In *Information Fusion (Fusion), 2015 18th International Conference on*, pages 1100–1105. IEEE, 2015.

[49] O. S. d. SOUSA JUNIOR et al. Um framework para suportar de forma semiautomática a atividade de desenvolvimento de software para mapreduce utilizando mde. 2017.

[50] G. H. Travassos, D. Gurov, and E. Amaral. *Introdução à engenharia de software experimental*. UFRJ, 2002.

[51] A. Van Deursen, P. Klint, and J. Visser. Domain-specific languages: An annotated bibliography. *ACM Sigplan Notices*, 35(6):26–36, 2000.

[52] T. R. VAZ et al. Cepfid: Plataforma de middleware para processamento de eventos complexos rfid. 2011.

[53] M. Voelter, S. Benz, C. Dietrich, B. Engelmann, M. Helander, L. C. Kats, E. Visser, and G. Wachsmuth. *DSL engineering: Designing, implementing and using domain-specific languages*. dslbook. org, 2013.

[54] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén. *Experimentation in software engineering*. Springer Science & Business Media, 2012.

[55] Y. Zheng, L. Capra, O. Wolfson, and H. Yang. Urban Computing: Concepts, Methodologies, and Applications. *ACM Transactions on Intelligent Systems and Technology*, 5(3):38:1—-38:55, sep 2014.

Apêndices

A Artefato Gerado pelo Cenário 5.1

```
1  public class Mobcons {
2      public static void main(String[] args) {
3          MobconsCore mobconsCore = new MobconsCore();
4          mobconsCore.init(true);
5                  /*
6                  * Rule Description: Regra Pátio Minério
7                  * Rule ID: 1
8                  */
9                  SpeedRestriction restriction11 = new SpeedRestriction(40.0);
10                 AreaContext contextPolygonalArea11 = new AreaContext(
11                     new Point2D.Double(10.0,10.0),
12                     new Point2D.Double(20.0,20.0),
13                     new Point2D.Double(30.0,30.0),
14                     new Point2D.Double(40.0,40.0));
15                 ConstraintTimeClause weekTimeCondition11 =
16                     ConstraintTimeClause.timeIntervalAndDaysOfWeekInArray(
17                     "07:00:00", "12:00:00",2,3,4,5,6);
18                 mobconsCore.newMC(restriction11,contextPolygonalArea11,weekTimeCondition11);
19                 /*
20                 * Rule Description: Regra Fórmula 1
21                 * Rule ID: 2
22                 */
23                 SpeedRestriction restriction22 = new SpeedRestriction(60.0);
24                 GroupUMContext contextGroup12 = new GroupUMContext(1,2,3,4,5);
25                 mobconsCore.newMC(restriction22,contextGroup12);
26                 /*
27                 * Rule Description: Regra Permanencia Pátio
28                 * Rule ID: 3
29                 */
30                 AreaRestriction restriction13 = new AreaRestriction(true,
31                                         new Point2D.Double(10.0,10.0),
32                                         new Point2D.Double(20.0,20.0),
33                                         new Point2D.Double(30.0,30.0),
34                                         new Point2D.Double(40.0,40.0));
35                 KindOfMUContext contextKindOfMU13 = new KindOfMUContext("Escavadeiras");
36                 mobconsCore.newMC(restriction13,contextKindOfMU13);
37     } //end of the method main
38 } //end of the class
```

B Artefato Gerado pelo Cenário 5.2

```
1  public class Mobcons {
2      public static void main(String[] args) {
3          MobconsCore mobconsCore = new MobconsCore();
4          mobconsCore.init(true);
5                  /*
6                  * Rule Description: Regra Velocidade
7                  * Rule ID: 1
8                  */
9                  SpeedRestriction restriction11 = new SpeedRestriction(60.0);
10                 SingleUMContext contextMU11 = new SingleUMContext(1);
11                 mobconsCore.newMC(restriction11,contextMU11);
12                 SingleUMContext contextMU21 = new SingleUMContext(2);
13                 mobconsCore.newMC(restriction11,contextMU21);
14                 SingleUMContext contextMU31 = new SingleUMContext(3);
15                 mobconsCore.newMC(restriction11,contextMU31);
16                 SingleUMContext contextMU41 = new SingleUMContext(4);
17                 mobconsCore.newMC(restriction11,contextMU41);
18                 /*
19                 * Rule Description: Regra Rota
20                 * Rule ID: 3
21                 */
22                 PathRestriction restriction13 = new PathRestriction(0.0,
23                                 new Point2D.Double(10.0,10.0),
24                                 new Point2D.Double(20.0,20.0),
25                                 new Point2D.Double(30.0,30.0),
26                                 new Point2D.Double(40.0,40.0),
27                                 new Point2D.Double(50.0,50.0));
28                 SingleUMContext contextMU13 = new SingleUMContext(1);
29                 mobconsCore.newMC(restriction13,contextMU13);
30                 SingleUMContext contextMU23 = new SingleUMContext(2);
31                 mobconsCore.newMC(restriction13,contextMU23);
32                 SingleUMContext contextMU33 = new SingleUMContext(3);
33                 mobconsCore.newMC(restriction13,contextMU33);
34                 SingleUMContext contextMU43 = new SingleUMContext(4);
35                 mobconsCore.newMC(restriction13,contextMU43);
36                 /*
37                 * Rule Description: Regra Distância 1 e 2
38                 * Rule ID: 2
39                 */
40                 DistanceRestriction restriction12 = new DistanceRestriction(new SingleUMContext(1),3.0, false);
41                 SingleUMContext contextMU22 = new SingleUMContext(2);
42                 mobconsCore.newMC(restriction12,contextMU22);
43                 /*
44                 * Rule Description: Regra Distância 2 e 3
45                 * Rule ID: 6
46                 */
47                 DistanceRestriction restriction16 = new DistanceRestriction(new SingleUMContext(2),3.0, false);
48                 SingleUMContext contextMU36 = new SingleUMContext(3);
49                 mobconsCore.newMC(restriction16,contextMU36);
50                 /*
51                 * Rule Description: Regra Distância 3 e 4
52                 * Rule ID: 8
53                 */
54                 DistanceRestriction restriction18 = new DistanceRestriction(new SingleUMContext(3),3.0, false);
55                 SingleUMContext contextMU48 = new SingleUMContext(4);
56                 mobconsCore.newMC(restriction18,contextMU48);
57     } //end of the method main
58 } //end of the class
```

C Artefato Gerado pelo Cenário 5.3

```
1  public class Mobcons {
2      public static void main(String[] args) {
3          MobconsCore mobconsCore = new MobconsCore();
4          mobconsCore.init(true);
5                  /*
6                  * Rule Description: Regra Velocidade
7                  * Rule ID: 1
8                  */
9                  SpeedRestriction restriction11 = new SpeedRestriction(40.0);
10                 SingleUMContext contextMU11 = new SingleUMContext(1);
11                 mobconsCore.newMC(restriction11,contextMU11);
12                 SingleUMContext contextMU21 = new SingleUMContext(2);
13                 mobconsCore.newMC(restriction11,contextMU21);
14                 SingleUMContext contextMU31 = new SingleUMContext(3);
15                 mobconsCore.newMC(restriction11,contextMU31);
16                 SingleUMContext contextMU41 = new SingleUMContext(4);
17                 mobconsCore.newMC(restriction11,contextMU41);
18                 SingleUMContext contextMU51 = new SingleUMContext(5);
19                 mobconsCore.newMC(restriction11,contextMU51);
20                 SingleUMContext contextMU61 = new SingleUMContext(6);
21                 mobconsCore.newMC(restriction11,contextMU61);
22                 /*
23                 * Rule Description: Regra Permanência Itaqui-Bacanga
24                 * Rule ID: 2
25                 */
26                 AreaRestriction restriction12 = new AreaRestriction(true,new Point2D.Double(400.0,400.0),10.0);
27                 GroupUMContext contextGroup12 = new GroupUMContext(1,2);
28                 ConstraintTimeClause weekTimeCondition12 =
29                     ConstraintTimeClause.timeIntervalAndDaysOfWeekInArray("06:00:00",   "18:00:00",2,3,4,5,6);
30                 mobconsCore.newMC(restriction12,contextGroup12,weekTimeCondition12);
31                 ConstraintTimeClause weekTimeCondition22 =
32                     ConstraintTimeClause.timeIntervalAndDaysOfWeekInArray("10:00:00",   "22:00:00",1,7);
33                 mobconsCore.newMC(restriction12,contextGroup12,weekTimeCondition22);
34                 /*
35                 * Rule Description: Regra Permanência Centro
36                 * Rule ID: 3
37                 */
38                 AreaRestriction restriction23 = new AreaRestriction(true,new Point2D.Double(10.0,10.0),5.0);
39                 GroupUMContext contextGroup23 = new GroupUMContext(3,4);
40                 ConstraintTimeClause weekTimeCondition33 =
41                     ConstraintTimeClause.timeIntervalAndDaysOfWeekInArray("06:00:00",    "18:00:00",2,3,4,5,6);
42                 mobconsCore.newMC(restriction13,contextGroup23,weekTimeCondition33);
43                 ConstraintTimeClause weekTimeCondition43 =
44                     ConstraintTimeClause.timeIntervalAndDaysOfWeekInArray("10:00:00",  "22:00:00",1,7);
45                 mobconsCore.newMC(restriction13,contextGroup23,weekTimeCondition43);
46                 /*
47                 * Rule Description: Regra Permanência Praias
48                 * Rule ID: 4
49                 */
50                 AreaRestriction restriction34 = new AreaRestriction(true,new Point2D.Double(1000.0,1000.0),20.0);
51                 GroupUMContext contextGroup34 = new GroupUMContext(6,5);
52                 ConstraintTimeClause weekTimeCondition64 =
53                     ConstraintTimeClause.timeIntervalAndDaysOfWeekInArray("06:00:00",   "18:00:00",2,3,4,5,6);
54                 mobconsCore.newMC(restriction14,contextGroup34,weekTimeCondition64);
55                 ConstraintTimeClause weekTimeCondition74 =
56                     ConstraintTimeClause.timeIntervalAndDaysOfWeekInArray("10:00:00",   "22:00:00",1,7);
57                 mobconsCore.newMC(restriction14,contextGroup34,weekTimeCondition74);
58     } //end of the method main
59 } //end of the class
```

D Artefato Gerado pelo Cenário 5.4

```
1  public class Mobcons {
2      public static void main(String[] args) {
3          MobconsCore mobconsCore = new MobconsCore();
4          mobconsCore.init(true);
5                  /*
6                  * Rule Description: Regra Velocidade Área do Píer
7                  * Rule ID: 1
8                  */
9                  SpeedRestriction restriction11 = new SpeedRestriction(10.0);
10                 AreaContext contextRectangularArea11 = new AreaContext(new
11                     Point2D.Double(100.0,100.0),
12                     new Point2D.Double(200.0,200.0));
13                 mobconsCore.newMC(restriction11,contextRectangularArea11);
14                 /*
15                 * Rule Description: Regra Acesso Visitantes
16                 * Rule ID: 2
17                 */
18                 AreaRestriction restriction12 = new AreaRestriction(false,new
19                     Point2D.Double(100.0,100.0),
20                     new Point2D.Double(200.0,200.0));
21
22                 KindOfMUContext contextKindOfMU32 = new KindOfMUContext("Visitantes");
23                 mobconsCore.newMC(restriction12,contextKindOfMU32);
24                 /*
25                 * Rule Description: Regra Velocidade Navios
26                 * Rule ID: 4
27                 */
28                 SpeedRestriction restriction24 = new SpeedRestriction(10.0);
29                 KindOfMUContext contextKindOfMU14 = new KindOfMUContext("Navios");
30                 mobconsCore.newMC(restriction24,contextKindOfMU14);
31                 /*
32                 * Rule Description: Regra Distância Mínima Navios
33                 * Rule ID: 5
34                 */
35                 DistanceRestriction restriction15 = new DistanceRestriction(new KindOfMUContext("Navios"),1.0, false);
36                 KindOfMUContext contextKindOfMU25 = new KindOfMUContext("Navios");
37                 mobconsCore.newMC(restriction15,contextKindOfMU25);
38     } //end of the method main
39 } //end of the class
```

E Artefato Gerado pelo Cenário 5.5

```
1  public class Mobcons {
2      public static void main(String[] args) {
3          MobconsCore mobconsCore = new MobconsCore();
4          mobconsCore.init(true);
5                  /*
6                  * Rule Description: Regra Velocidade Área de Manobras
7                  * Rule ID: 1
8                  */
9                  SpeedRestriction restriction11 = new SpeedRestriction(20.0);
10                 AreaContext contextRectangularArea11 = new AreaContext(
11                     new Point2D.Double(100.0,100.0),
12                     new Point2D.Double(200.0,200.0));
13                 mobconsCore.newMC(restriction11,contextRectangularArea11);
14                 /*
15                 * Rule Description: Regra Acesso Passageiros
16                 * Rule ID: 2
17                 */
18                 AreaRestriction restriction12 = new AreaRestriction(
19                     false,new Point2D.Double(100.0,100.0),
20                     new Point2D.Double(200.0,200.0));
21                 KindOfMUContext contextKindOfMU12 = new KindOfMUContext("Passageiros");
22                 mobconsCore.newMC(restriction12,contextKindOfMU12);
23                 /*
24                 * Rule Description: Regra Acesso Veículos Carga
25                 * Rule ID: 3
26                 */
27                 AreaRestriction restriction13 = new AreaRestriction(
28                     false,new Point2D.Double(100.0,100.0),
29                     new Point2D.Double(200.0,200.0));
30                 KindOfMUContext contextKindOfMU13 = new KindOfMUContext("Veiculos Carga");
31                 mobconsCore.newMC(restriction13,contextKindOfMU13);
32                 /*
33                 * Rule Description: Regra Velocidade Área de Manobras Dias Especiais
34                 * Rule ID: 4
35                 */
36                 SpeedRestriction restriction24 = new SpeedRestriction(10.0);
37                 AreaContext contextRectangularArea14 = new AreaContext(
38                     new Point2D.Double(100.0,100.0),
39                     new Point2D.Double(200.0,200.0));
40                 ConstraintTimeClause dateCondition14 = ConstraintTimeClause.date(-1,12,31);
41                 mobconsCore.newMC(restriction24,contextRectangularArea14,dateCondition14);
42                 ConstraintTimeClause dateCondition24 = ConstraintTimeClause.date(-1,12,25);
43                 mobconsCore.newMC(restriction24,contextRectangularArea14,dateCondition24);
44     } //end of the method main
45 } //end of the class
```

F Planilha de Perfil dos Participantes

Participantes							
	Nome	Formação Acadêmica	Titulação Máxima	Atuação Profissional	Desenvolvimento?	MDE?	Eclipse?
1	Italo Tiago Gomes Souza	Ciência da Computação	Graduando	Estudante	x		x
2	Tércio Santana Silva Sousa	Ciência da Computação	Mestrando	Professor	x	x	x
3	Marcelino Mendes da Silva Neto	Ciência da Computação	Mestrando	Estudante	x		x
4	Anderson Soares Costa	Ciência da Computação	Mestrando	Estudante	x		x
5	Sebastião Carneige	Ciência da Computação	Graduando	Analista de Sistemas	x	x	x
6	Lucas Rodrigues Santos	Administração	Graduando	Estudante			
7	Kelly Rebeca da S. Costa	Ciência da Computação	Graduando	Estudante	x		
8	Lucas Cunha de Carvalho	Ciência da Computação	Graduando	Estudante	x		x
9	Fernando Henrique F. Leite	Ciência da Computação	Graduando	Estudante	x		x
10	Cailini da Silva Lima	Direito	Graduado	Servidora Pública			
11	Fernando Benedito Veras Magalhães	Ciência da Computação	Graduado	Estudante	x		x
12	Samuel Felipe Dias de Carvalho	Ciência da Computação	Graduando	Estudante	x		
13	Eduardo Devidson Costa Bezerra	Ciência da Computação	Doutorando	Analista de Sistemas	x	x	x
14	Helvio Oliveira Carvalho	Ciência da Computação	Graduado	Analista de Sistemas	x	x	x
15	Luiz Gonzaga de Albuquerque Neto	Ciência da Computação	Graduado	Analista de Sistemas	x		x
16	Wellington da Silva Moraes	Administração	Pós-graduação	DBA	x	x	x
17	Valdeci Ribeiro da Silva Junior	Ciência da Computação	Pós-graduação	Analista de Sistemas	x	x	x
18	Jadson do Nascimento dos Santos	Ciência da Computação	Graduado	Analista de Sistemas	x	x	x
19	Mércio Pio de Carvalho	Ciência da Computação	Graduado	Analista de Sistemas	x	x	x
20	Egídio de Carvalho Ribeiro Junior	Ciência da Computação	Mestre	Analista de Sistemas	x	x	x
21	Osvaldo Silva de Sousa Junior	Ciência da Computação	Doutor	Analista de Sistemas	x	x	x
22	Samira Alves Brito	Direito	Graduado	Servidora Pública			
23	Rosângela Pinho de Miranda	Química	Mestre	Servidora Pública			
24	Fernando Rocha Nogueira	Técnico	Nível Médio	Assistente Odontológico			
25	Camila da Silva Lima	Ciências Contábeis	Pós-graduação	Analista de Qualidade			
26	Thayna Deva Santos Moura	Direito	Graduado	Advogada			
27	Carmosina Pereira da Silva	Administração	Graduado	Microempresária			
28	Giulliana Vasconcelos Rios	Direito	Graduado	Servidora Pública			
29	Suze Brito	Nutrição	Graduado	Nutricionista			
30	Jonas Faustino	Engenharia	Graduado	Engenheiro			

	FA	FR
Ciência da Computação	18	60.0%
Administração	3	10.0%
Direito	4	13.3%
Química	1	3.3%
Ciências Contábeis	1	3.3%
Técnico	1	3.3%
Nutrição	1	3.3%
Engenharia	1	3.3%
TOTAL	30	100%

	FA	FR
Nível Médio	1	3.3%
Graduando	7	23.3%
Graduado	12	40.0%
Pós-graduação	3	10.0%
Mestrando	3	10.0%
Mestre	2	6.7%
Doutorando	1	3.3%
Doutor	1	3.3%
TOTAL	30	100%

	FA	FR
Desenvolvimento	19	63.3%
MDE	10	33.3%
Eclipse	17	56.7%

FA – Frequência Absoluta
FR – Frequência Relativa

	FA	FR
Profissionais	21	70%
Estudante	9	30%
TOTAL	30	100%

Figura 1: Planilha de perfil dos participantes.

G Planilha da Avaliação de Esforço

Tempo de Modelagem dos Cenários Por Participante (hh:mm:ss)			
Partipante	Fácil	Médio	Difícil
1	00:03:39	00:02:24	00:17:04
2	00:08:35	00:12:07	00:15:40
3	00:07:28	00:04:20	00:10:20
4	00:04:40	00:01:47	00:14:25
5	00:06:00	00:05:15	00:25:00
6	00:02:56	00:11:00	00:11:23
7	00:05:00	00:12:45	00:12:00
8	00:04:58	00:06:39	00:17:43
9	00:05:16	00:03:40	00:16:25
10	00:03:30	00:07:00	00:10:00
11	00:05:00	00:05:00	00:15:00
12	00:04:50	00:07:30	00:16:16
13	00:03:28	00:08:18	00:23:30
14	00:02:00	00:02:00	00:15:00
15	00:05:00	00:10:00	00:45:00
16	00:10:00	00:11:12	00:40:00
17	00:05:20	00:05:00	00:30:00
18	00:07:25	00:10:04	00:30:00
19	00:06:10	00:07:20	00:30:23
20	00:03:00	00:05:00	00:15:00
21	00:03:20	00:10:00	00:13:00
22	00:04:12	00:08:27	00:29:15
23	00:08:43	00:05:00	00:20:00
24	00:03:56	00:05:00	00:15:36
25	00:02:00	00:07:23	00:28:00
26	00:11:25	00:07:16	00:52:00
27	00:10:41	00:09:00	00:32:00
28	00:06:00	00:05:17	00:28:42
29	00:09:12	00:05:00	00:38:00
30	00:02:00	00:03:21	00:19:53

Em Minutos		
Fácil	Médio	Difícil
3.65	2.40	17.07
8.58	12.12	15.67
7.47	4.33	10.33
4.67	1.78	14.42
6.00	5.25	25.00
2.93	11.00	11.38
5.00	12.75	12.00
4.97	6.65	17.72
5.27	3.67	16.42
3.50	7.00	10.00
5.00	5.00	15.00
4.83	7.50	16.27
3.47	8.30	23.50
2.00	2.00	15.00
5.00	10.00	45.00
10.00	11.20	40.00
5.33	5.00	30.00
7.42	10.07	30.00
6.17	7.33	30.38
3.00	5.00	15.00
3.33	10.00	13.00
4.20	8.45	29.25
8.72	5.00	20.00
3.93	5.00	15.60
2.00	7.38	28.00
11.42	7.27	52.00
10.68	9.00	32.00
6.00	5.28	28.70
9.20	5.00	38.00
2.00	3.35	19.88

Tempo Médio		
5.52	6.80	22.89

Desvio Padrão		
1.17	0.67	1.99

Tempo Médio de Modelagem dos Cenários Por Especialistas		
Fácil	Médio	Difícil
00:01:00	00:01:30	00:04:00

	Fácil	Médio	Difícil
Es	1	1.5	4
Pa	5.52	6.80	22.89

	Tempo Médio	Desvio Padrão
Fácil	5.52	1.16
Médio	6.8	0.67
Difícil	22.88	1.99

Modelagens Corretas Por Participante			
Participante	Fácil	Médio	Difícil
1	Sim	Sim	Parcialmente
2	Sim	Sim	Não
3	Sim	Sim	Parcialmente
4	Sim	Sim	Sim
5	Sim	Sim	Parcialmente
6	Sim	Parcialmente	Não
7	Sim	Sim	Não
8	Sim	Sim	Parcialmente
9	Sim	Sim	Parcialmente
10	Sim	Sim	Parcialmente
11	Sim	Sim	Sim
12	Sim	Sim	Parcialmente
13	Sim	Sim	Parcialmente
14	Sim	Parcialmente	Parcialmente
15	Sim	Sim	Parcialmente
16	Sim	Não	Não
17	Sim	Sim	Parcialmente
18	Sim	Parcialmente	Parcialmente
19	Sim	Parcialmente	Parcialmente
20	Sim	Sim	Parcialmente
21	Sim	Sim	Sim
22	Parcialmente	Parcialmente	Não
23	Sim	Parcialmente	Não
24	Parcialmente	Parcialmente	Parcialmente
25	Parcialmente	Não	Parcialmente
26	Parcialmente	Parcialmente	Parcialmente
27	Sim	Sim	Parcialmente
28	Parcialmente	Parcialmente	Parcialmente
29	Sim	Parcialmente	Parcialmente
30	Sim	Parcialmente	Parcialmente

Fácil	FA	FR
Sim	25	83.3%
Não	0	0.0%
Parcialmente	5	16.7%
TOTAL	30	100%

Médio	FA	FR
Sim	17	56.7%
Não	2	6.7%
Parcialmente	11	36.7%
TOTAL	30	100%

Difícil	FA	FR
Sim	3	10.0%
Não	6	20.0%
Parcialmente	21	70.0%
TOTAL	30	100%

Figura 2: Planilha da avaliação de esforço.

H Planilha da Avaliação de Percepção da Qualidade

PSSUQ																	
	Avaliação Geral do Sistema (**OVERALL**)																
	Utilidade do Sistema (**SYSUSE**)						Qualidade Informação (**INFOQUAL**)						Qualidade Interface (**INTERQUAL**)				
Participante	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**	**Q6**	**Q7**	**Q8**	**Q9**	**Q10**	**Q11**	**Q12**	**Q13**	**Q14**	**Q15**	**Q16**	
1	1	1	2	1	2	1	3	1	2	2	2	1	1	1	1	1	
2	1	2	2	1	1	1	3	2	1	1	1	1	1	1	1	1	
3	2	3	3	3	2	2	4	4	3	4	4	2	3	3	2	2	
4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
5	2	2	3	1	2	1	1	2	2	4	2	2	2	2	2	2	
6	3	1	3	4	1	1	1	1	1	1	3	3	4	4	1	1	
7	2	2	3	4	2	2	4	4	2	2	2	4	4	3	4	1	
8	1	2	2	2	2	2	2	2	1	2	2	2	2	2	2	2	
9	5	4	2	4	1	1	7	4	1	1	1	7	7	4	1	4	
10	1	1	4	2	1	4	2	1	1	5	4	2	1	1	3	1	
11	1	1	1	1	1	1	2	1	2	1	1	2	1	1	1	1	
12	1	1	2	1	1	1	1	1	1	1	1	1	1	1	1	1	
13	2	1	1	1	1	2	4	2	1	1	1	2	1	1	2	1	
14	2	2	2	3	1	3	3	2	1	1	1	1	1	1	2	1	
15	2	2	3	2	1	1	3	3	4	3	3	2	1	1	1	1	
16	3	3	3	3	3	3	3	4	3	3	3	3	3	3	3	3	
17	1	1	1	2	1	1	1	1	2	4	4	2	2	1	1	1	
18	1	1	1	2	1	1	1	1	2	2	2	1	1	2	1	1	
19	2	2	2	1	1	2	1	2	1	2	2	2	2	1	1	1	
20	1	1	1	1	1	1	1	1	1	1	1	1	2	1	1	1	
21	2	2	2	2	2	1	4	3	3	3	2	2	2	2	4	2	
22	1	2	2	2	1	3	4	3	2	2	2	2	1	1	1	1	
23	2	2	3	2	2	2	3	3	3	2	2	2	2	2	2	2	
24	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
25	1	2	2	1	1	1	2	2	2	1	4	4	2	2	2	1	
26	3	3	2	2	1	3	3	4	4	4	3	3	5	4	2	2	
27	3	2	2	2	1	1	2	2	3	1	1	1	2	2	1	1	
28	1	1	1	2	1	1	2	2	1	1	2	2	1	1	1	1	
29	2	2	3	2	1	2	6	4	2	2	4	3	3	3	2	2	
30	1	1	1	1	2	2	2	2	2	2	2	1	2	2	2	2	
Qtde Resp	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**	**Q6**	**Q7**	**Q8**	**Q9**	**Q10**	**Q11**	**Q12**	**Q13**	**Q14**	**Q15**	**Q16**	
1	15	13	9	12	21	17	9	10	13	13	10	10	13	15	16	20	
2	10	13	12	12	8	8	7	10	10	9	11	13	10	8	10	8	
3	4	3	8	3	1	4	7	4	5	3	4	4	3	4	2	1	
4	0	1	1	3	0	1	5	6	2	4	5	2	2	3	2	1	
5	1	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	
6	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	
7	0	0	0	0	0	0	1	0	0	0	0	1	1	0	0	0	

OVERALL		
	FA	**FR**
1	216	45.0%
2	159	33.1%
3	60	12.5%
4	38	7.9%
5	3	0.6%
6	1	0.2%
7	3	0.6%
TOTAL	480	100%

SYSUSE		
	FA	**FR**
1	87	48.3%
2	63	35.0%
3	23	12.8%
4	6	3.3%
5	1	0.6%
6	0	0.0%
7	0	0.0%
TOTAL	180	100%

INFOQUAL		
	FA	**FR**
1	65	36.1%
2	60	33.3%
3	27	15.0%
4	24	13.3%
5	1	0.6%
6	1	0.6%
7	2	1.1%
TOTAL	180	100%

INTERQUAL		
	FA	**FR**
1	64	53.3%
2	36	30.0%
3	10	8.3%
4	8	6.7%
5	1	0.8%
6	0	0.0%
7	1	0.8%
TOTAL	120	100%

Overall	
Positivo	435
Neutro	38
Negativo	7
	480

SysUse	
Positivo	173
Neutro	6
Negativo	1
	180

InfoQual	
Positivo	152
Neutro	24
Negativo	4
	180

INTERQUAL	
Positivo	110
Neutro	8
Negativo	2
	120

Figura 3: Planilha da avaliação de percepção da qualidade.

I Questionário de Avaliação

Universidade Federal do Maranhão
Centro de Ciências Exatas de Tecnologia
Programa de Pós-Graduação em Ciência da Computação
Laboratório de Sistemas Distribuídos Inteligentes – Lsdi

Aluno: Adalberto Jr.
Orientador: Luciano Reis/Francisco Silva

Questionário de Avaliação da Qualidade

MobCons-AT: Uma Ferramenta de Autoria para Restrições de Mobilidade Baseada na Transformação de Modelos

1. Participante

Nome:_____

Email:_____

Formação Acadêmica: _____

Titulação Máxima: _____

Atuação Profissional: _____

2. Experiências

Desenvolvimento de softwares:	Sim( ) Não( )
Ferramentas MDE:	Sim( ) Não( )
IDE Eclipse:	Sim( ) Não( )

São Luís
2018

Figura 4: Questionário de avaliação - página 1.

3. Avaliação da Qualidade

Esta avaliação consiste em um processo com duas atividades onde será avaliado o nível de satisfação dos voluntários, com o objetivo de analisar se a ferramenta consegue estabelecer uma interface intuitiva, apresentando-se como um canal de fácil manipulação de especificação de restrições de mobilidade e fornecendo recursos suficientes para que tais especificações sejam programáveis em alto nível.

➔ Atividade 1: Modelar na MobCons-AT os cenários abaixo.

Nível: fácil
"Todos as viaturas policiais do Estado do Maranhão devem transitar a uma velocidade máxima de 60 km/h".

Tempo gasto: _____
Modelagem feita corretamente? Sim( ) Não( ) Parcialmente( )

Nível: médio
"Uma empresa de transporte coletivo deseja que todos os seus veículos do tipo "ônibus" transitem no centro da cidade a uma velocidade máxima de 40 km/h, de segunda a sexta, entre 8h e 18h".

Tempo gasto: _____
Modelagem feita corretamente? Sim( ) Não( ) Parcialmente( )

Nível: difícil
No Porto de Itaqui, por motivos de segurança, somente funcionários autorizados da área operacional e veículos de carga e descarga podem transitar pelos piers, cuja velocidade máxima para circulação de veículos é de 10 km/h. Os navios, quando ancorados no mar aguardando atracar, devem manter uma distância mínima de 1 km uns dos outros e ao se deslocarem para atracamento no píer, devem manter uma velocidade mínima de 3 km/h e máxima de 10 km/h

Tempo gasto: _____
Modelagem feita corretamente? Sim( ) Não( ) Parcialmente( )

Figura 5: Questionário de avaliação - página 2.

➔ Atividade 2: Preencher um questionário para avaliar a eficiência e usabilidade da ferramenta. Para isso utilizou-se o questionário **Post-Study System Usability Questionnaire** (**PSSUQ**), com adaptações, desenvolvido pela IBM para avaliação da satisfação do usuário e da usabilidade, largamente utilizado na indústria e academia.

Serão consideradas as notas de avaliação de cada quesito apresentado como **POSITIVO se menor igual três (3)**, **NEUTRO se igual a quatro (4)** e **NEGATIVO se maior igual a cinco (5)**, a levar-se em consideração avaliação psicrométrica baseada em escala de Linkert (LINKERT, 1932).

1. No geral, estou satisfeito com o quão fácil é usar o sistema.

Concordo Fortemente 1( ) 2( ) 3( ) 4( ) 5( ) 6( ) 7( ) Discordo Fortemente

2. Foi fácil utilizar este sistema.

Concordo Fortemente 1( ) 2( ) 3( ) 4( ) 5( ) 6( ) 7( ) Discordo Fortemente

3. Eu fui capaz de completar as tarefas e cenários de forma rápida, usando este sistema.

Concordo Fortemente 1( ) 2( ) 3( ) 4( ) 5( ) 6( ) 7( ) Discordo Fortemente

4. Eu me senti confortável usando este sistema.

Concordo Fortemente 1( ) 2( ) 3( ) 4( ) 5( ) 6( ) 7( ) Discordo Fortemente

5. Foi fácil aprender a usar este sistema.

Concordo Fortemente 1( ) 2( ) 3( ) 4( ) 5( ) 6( ) 7( ) Discordo Fortemente

6. Eu acredito que eu poderia me tornar produtivo rapidamente usando este sistema.

Concordo Fortemente 1( ) 2( ) 3( ) 4( ) 5( ) 6( ) 7( ) Discordo Fortemente

7. As mensagens de erros do sistema foram claras o suficiente para me ajudar na correção de erros.

Concordo Fortemente 1( ) 2( ) 3( ) 4( ) 5( ) 6( ) 7( ) Discordo Fortemente

8. Sempre que eu cometi algum erro, eu pude recuperar de forma fácil e rápida.

Concordo Fortemente 1( ) 2( ) 3( ) 4( ) 5( ) 6( ) 7( ) Discordo Fortemente

9. As informações (ajuda online, tela de mensagens e outros documentos) fornecidas com este sistema foram claras.

Concordo Fortemente 1( ) 2( ) 3( ) 4( ) 5( ) 6( ) 7( ) Discordo Fortemente

3

Figura 6: Questionário de avaliação - página 3.

10. Foi fácil encontrar a informação que eu precisava.

Concordo Fortemente 1( ) 2( ) 3( ) 4( ) 5( ) 6( ) 7( ) Discordo Fortemente

11. A informação foi eficaz em me ajudar a completar as tarefas e cenários.

Concordo Fortemente 1( ) 2( ) 3( ) 4( ) 5( ) 6( ) 7( ) Discordo Fortemente

12. A organização das informações nas telas do sistema é clara.

Concordo Fortemente 1( ) 2( ) 3( ) 4( ) 5( ) 6( ) 7( ) Discordo Fortemente

13. A interface deste sistema é agradável.

Concordo Fortemente 1( ) 2( ) 3( ) 4( ) 5( ) 6( ) 7( ) Discordo Fortemente

14. Eu gostei de usar a interface deste sistema.

Concordo Fortemente 1( ) 2( ) 3( ) 4( ) 5( ) 6( ) 7( ) Discordo Fortemente

15. Este sistema tem todas as funções e capacidades que eu esperava que ele tivesse.

Concordo Fortemente 1( ) 2( ) 3( ) 4( ) 5( ) 6( ) 7( ) Discordo Fortemente

16. No geral, estou satisfeito com este sistema.

Concordo Fortemente 1( ) 2( ) 3( ) 4( ) 5( ) 6( ) 7( ) Discordo Fortemente

4

Figura 7: Questionário de avaliação - página 4.