UNISINOS

Programa Interdisciplinar de Pós-Graduação em

Computação Aplicada

Mestrado Acadêmico

Vivian Teresinha Pedó da Silva

UML2Context:

Uma Extensão da UML para Modelagem de Contexto

São Leopoldo, 2015

A FICHA CATALOGRÁFICA DEVE SER ELABORADA POR BIBLIOTECÁRIO COM

REGISTRO NO CRB.

UNISINOS

UNIVERSIDADE DO VALE DO RIO DOS SINOS
Ciências Exatas e Tecnológicas
Programa Interdisciplinar de Pós-Graduação em Computação Aplicada – PIPCA

Vivian Pedó

UML2Context:
Uma Extensão da UML para Modelagem de Contexto

Dissertação apresentada como requisito parcial para a obtenção do título de Mestre, pelo Programa Interdisciplinar de Pós-Graduação em Computação Aplicada da Universidade do Vale do Rio dos Sinos – UNISINOS

Aprovado em 15 de junho de 2015

BANCA EXAMINADORA

Prof. Dr. André Rauber Du Bois – Universidade Federal Pelotas

Prof. Dr. Cristiano André da Costa – UNISINOS

Prof. Dr. Jorge Luis Victória Barbosa – UNISINOS

Prof. Dr. Kleinner Silva Farias de Oliveira – UNISINOS

Prof. Dr. Cristiano André da Costa (Orientador)
Prof. Dr. Kleinner Silva Farias de Oliveira (Coorientador)

Visto e permitida a impressão
São Leopoldo,

Prof. Dr. Cristiano André da Costa
Coordenador PPG em Computação Aplicada

A Deus em primeiro lugar, aos meus pais e ao meu marido por me apoiarem e por não me deixarem desistir a cada vez que eu não tinha mais forças para continuar. Aos meus orientadores que me passaram desafios para que desse o meu melhor, que não desistiram de mim e que confiaram em mim. Sem vocês todos eu não teria finalizado essa batalha.

AGRADECIMENTOS

Agradeço a todas as pessoas e instituições que me apoiaram de uma forma ou de outra.

Aos meus pais Marli Pedó e Jorge Coelho, e ao meu marido Alexandre Rodrigues que me apoiaram e acreditaram em mim sempre.

Ao Cristiano André da Costa, pelo apoio, incentivo, orientação e confiança.

Ao Kleinner Silva Farias de Oliveira pelo apoio, incentivo, orientação e confiança.

Aos professores que me apoiaram desde o início dessa jornada e me indicaram para o mestrado, professor Marcelo Henrique Euzébio Batista e Letícia Silva Garcia.

À minha chefe Márcia Scheid por me incentivar a iniciar o mestrado e por me ajudar me liberando quando precisei e a empresa que trabalho CWI pelo apoio quando mais precisei.

A minha professora de inglês Carolina Girardi, pelo apoio e as aulas focadas no meu mestrado.

Aos amigos e colegas pelas indicações e apoio Tiago André Jost e Rodrigo Reis.

À Unisinos por me acolher como estudante e por ser esta instituição de excelência.

Ao CAPES pela concessão da bolsa de estudos.

Aos professores da UFRGS, Fernanda Lima Kastensmidt e Leila Ribeiro pela oportunidade de iniciar o mestrado na instituição como aluna especial.

A todos os professores e funcionários do PIPCA.

“Em tempos em que quase ninguém se olha nos olhos, em que a maioria das pessoas pouco se interessa pelo que não lhe diz respeito, só mesmo agradecendo àqueles que percebem nossas descrenças, indecisões, suspeitas, tudo o que nos paralisa, e gastam um pouco da sua energia conosco, insistindo. ” (Obrigada por Insistir de Martha Medeiros)

RESUMO

Pesquisadores e profissionais da indústria reconhecem a importância do desenvolvimento de sistemas cientes de contexto, visto que tais sistemas cada vez mais fazem parte da vida cotidiana das pessoas. Para projetar, comunicar decisões de projeto e representar os aspectos estruturais e comportamentais destes sistemas, os desenvolvedores comumente utilizam a Linguagem de Modelagem Unificada (UML), a qual é amplamente reconhecida como linguagem padrão de modelagem de software. Porém, dada a crescente diversidade de técnicas de modelagem e a incapacidade da UML em representar os conceitos de sistemas cientes de contexto, é particularmente desafiante para os desenvolvedores objetivamente projetar e comunicar decisões de projetos de tais sistemas. Consequentemente, os desenvolvedores geralmente não dispõem de ferramentas que facilitem a representação de contexto de uma forma adequada, ao passo que propõem notações para contornar o problema, agravando ainda mais o problema da heterogeneidade das técnicas de modelagem atuais. O problema central é que a UML é imprecisa para representar o conceito de contexto e rígida para incorporar novos conceitos. Outro problema é que as ferramentas acadêmicas e comerciais de modelagem atuais - incluindo IBM RSA, Astah, Borland Together - não dão suporte à representação e à validação de modelos de contexto. Este trabalho, portanto, propõe uma extensão da UML para modelagem de contexto, a qual visa não só permitir a representação e validação dos principais aspectos de contexto, como também potencializar uma melhor comunicação de decisões de projeto de sistemas cientes de contexto. Também é proposto um ambiente de modelagem específico de domínio para tornar possível a modelagem de sistemas cientes de contexto, o qual foi implementado como um *plug-in* da plataforma Eclipse utilizando os frameworks GMF, EMF e UML2tool. Este ambiente permite não só representar o conceito de contexto seguindo a abordagem proposta, bem como avaliar a corretude dos modelos criados. A extensão e a ferramenta proposta foram avaliadas através de um questionário abordando a modelagem proposta comparada com modelagens que utilizam a UML pura. Essa avaliação permitiu determinar os reais benefícios do trabalho desenvolvido e onde os resultados sugerem que a modularização das informações de contexto em um novo conceito chamado de UML2Context traz benefícios, quando comparada com a decomposição de tais informações com a UML Pura. Os resultados apontaram que a UML2Context aumentou a taxa de respostas corretas em 28,41%, reduziu o esforço de interpretação em 61,03% e melhorou a interpretação dos modelos de contexto em 35,98%, se mostrando eficaz para modelagem de sistemas ubíquos.

Palavras-Chaves: Modelagem de Sistemas. Metamodelo. UML. Ciência de Contexto. Computação Móvel. Computação Ubíqua.

ABSTRACT

Researchers and industry professionals recognize the importance of developing context-aware systems, as these systems increasingly are part of everyday life of people. To design, communicate design decisions and represent the structural and behavioral aspects of these systems, developers commonly use the Unified Modeling Language (UML), which is widely recognized as the standard language of software modeling. However, given the increasing diversity of modeling techniques and UML's inability to represent the concepts of context-aware systems, is particularly challenging for developers to design and objectively communicate design decisions of such systems. Usually developers don't have the tools to facilitate the representation of context in an appropriate manner, whereas propose notations to solve the problem, further aggravating the problem of heterogeneity of current modeling techniques. The main problem is that UML is inaccurate to represent the concept of context and rigid to incorporate new concepts. Another problem is that academic and commercial tools of current modeling - including IBM RSA, Astah, Borland Together - do not support the representation and validation of context models. This paper therefore proposes an extension of UML for modeling context, which aims not only to allow the representation and validation of the key aspects of context, but also enhance communication of project decision of context aware systems. It also proposes a domain specific modeling environment that enable the modeling of context-aware systems, which was implemented as an Eclipse platform plugin using the GMF frameworks, EMF and UML2tool. This environment will not only represent the concept of context following the proposed approach and to evaluate the correctness of the models. The extent and the proposed tool was evaluated through a questionnaire addressing the proposed model compared with modelings that using the standard UML. This avaliation allowed us to evaluate the real benefits of work and where the results suggest that the modularization of context information in a new concept called UML2Context brings benefits compared with the decomposition of such information with the pure UML. The results link that UML2Context increased the rate of correct answers in 28.41%, reduced the effort of interpretation in 61,03% and improved the interpretation of context models in 35.98%, proving effective for modeling ubiquitous systems.

Keywords: Systems Modeling. Metamodel. UML. Context-Aware. Mobile Computing. Ubiquous Computing.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

CWM	Common Warehouse Metamodel
EMF	Eclipse Metamodel Framework
EMOF	Essencial Meta-Object Facility
GEF	Graphical Eclipse Framework
GMF	Graphical Modeling Framework
MDA	Model Driven Architecture
MOF	Meta Object Facility
OO	Object Orientation
OCL	Object Constraint Language
OMG	Object Management Group
ORM	Object Role Modeling
PIPCA	Programa Interdisciplinar de Pós-Graduação em Computação Aplicada
TAO	Taming Agents and Objects
TIER	Padrão para mensurar o nível de exigência de uma infraestrutura
UML	Unified Modeling Language
UML CAProf	UML Context-Aware Profile

SUMÁRIO

1 INTRODUÇÃO

Sistemas de informação móveis e ubíquos vêm ganhando atenção significativa em várias áreas devido ao rápido desenvolvimento de uma ampla gama de tecnologias móveis e sem fio, como WiMax, 3G e 4G. Computação móvel, ubíqua (ou pervasivas) são relevantes aos sistemas de informação devido ao fato de que estes conceitos tecnológicos impactam profundamente nas formas de fazer as coisas por parte das empresas e dos indivíduos (AHLUWALIA et al., 2014). O uso de dispositivos portáteis leva ao conceito de computação móvel que pode ser resumida como "informação na pontas dos dedos a qualquer momento e em qualquer lugar" (SATYANARAYANAN, 2010). Estas classes de sistemas, necessitam que sejam desenvolvidas aplicações mais robustas, explorando a mobilidade do usuário e a natureza dinâmica das infraestruturas mais modernas de computação (COSTA et al., 2010; DIX et al., 2000). Mark Weiser, em seu famoso artigo *The Computer for 21th Century* (WEISER, 1991), diz que a computação ubíqua compreende a ideia de integrar os computadores perfeitamente no mundo fazendo assim com que a tecnologia e sua complexidade desapareça. Estas aplicações devem estar cientes de seus contextos de interesse e se adaptar às mudanças que ocorrem neles. Entretanto o desenvolvimento de aplicações que interpretem o ambiente, se adaptem ao contexto e permaneçam funcionando enquanto o usuário se locomove, ainda é um dos grandes desafios de pesquisa na área (COSTA et al., 2012 e LOPES et al., 2013).

Neste cenário de desafios, uma das principais questões abordadas na área de computação móvel e ubíqua é a forma de expressar contexto. Um sistema ciente de contexto requer um modelo de contexto apropriado para representar e manipular informações adquiridas do contexto onde ele se encontra. Para tentar captar as necessidades desse conceito utilizam-se tecnologias para suportar o processamento semântico, como ontologias que são um instrumento promissor para especificar conceitos e inter-relações, e que são adequados para apoiar os serviços de raciocínio no âmbito das aplicações ubíquas (COSTA et al., 2012; STRANG e LINNHOFF-POPIEN, 2004; GU et al., 2004). Assim procuramos nas ontologias o vocabulário específico para maior compreensão do contexto.

No entanto, a construção de sistemas cientes de contexto é uma tarefa complexa, pois sem a intervenção do usuário é preciso adaptar seu comportamento às mudanças de contexto, através da combinação de informações de várias fontes diferentes. Um dos meios utilizados são técnicas de engenharia de software para lidar com sua complexidade, como técnicas de modelagem e frameworks baseados em soluções específicas (HOYOS; GARCÍA-MOLINA; BOTÍA, 2013). Devido a essa complexidade é importante que, na medida do possível, seja facilitado o trabalho dos desenvolvedores, pois os sistemas cientes de contexto têm crescido com os avanços tecnológicos, necessitando cada vez mais de softwares que suportem esses conceitos. Nesses softwares o tratamento do contexto em que a aplicação vai atuar é o ponto chave para o seu funcionamento. Assim, vários esforços no sentido de entender e categorizar o contexto vêm sendo pesquisados com essa finalidade, tais como Gu (2004), Bulcão (2006), Costa et al. (2012) Lopes et al. (2013) e Benselim e Seridi-Bouchelaghem (2013). Vemos nestes artigos que a criação de modelos auxilia na compreensão e raciocínio sobre o conceito de contexto, tendo em vista que um entendimento melhor de como o contexto pode ser usado, vai ajudar os desenvolvedores a determinar os comportamentos das aplicações, tornando-as ciente de contexto (DEY, 2001; HOYOS; GARCÍA-MOLINA; BOTÍA, 2013). Além disso, analisando muitas aplicações hoje nota-se que diversas ferramentas acadêmicas e comerciais de modelagem atuais como por exemplo IBM RSA[1], Astah[2], Borland Together[3], não dão suporte à representação e à validação de modelos de contexto de forma clara.

[1] IBM RSA: http://www.ibm.com/developerworks/br/downloads/r/architect/
[2] Astah: http://astah.net/
[3] Borland Together: https://www.borland.com/products/together/

Sendo assim para facilitar o trabalho dos desenvolvedores é proposto uma linguagem de modelagem para sistemas cientes de contexto associada a uma ferramenta que seja capaz de expressar particularidades que são encontradas ao longo do desenvolvimento destes sistemas. Particularidades tais como informações sobre os usuários do sistema, dispositivos que serão utilizados, localização física da aplicação, atividade exercida pela aplicação, dentre outras atividades e assim buscaram aprofundar a modelagem para os conceitos utilizados nessa área de agentes inteligentes, assim como também buscaram a UML para a modelagem. Os autores afirmam que é mais fácil aceitar um novo paradigma se os desenvolvedores e engenheiros de software conseguem usá-lo com um esforço mínimo. Como esta é uma linguagem conhecida e comum é possível a reutilização de componentes já existentes, bem como o conhecimento que já se possui. Pode-se observar em Silva, Choren e Lucena (2008), Benselim e Seridi-Bouchelaghem (2013) e Sun, France e Ray (2013) que são importantes os esforços para estender as técnicas de notação e de modelagem tradicionais para uso específico em cada área.

Pode-se concluir que para os desenvolvedores de software seria mais fácil trabalhar com uma modelagem que permita representar de forma efetiva tais particularidades encontradas tipicamente em sistemas cientes de contexto. De acordo com Fowler e Scotts (2000), desenvolvedores utilizam modelos, incluindo diagramas UML, para representar aspectos estruturais e comportamentais que são essenciais durante a fase de análise, projeto, desenvolvimento e implantação de sistemas. A criação de modelos auxilia na compreensão e raciocínio sobre contextos (HOYOS; GARCÍA-MOLINA; BOTÍA, 2013). A UML também proporciona uma melhor comunicação para decisões de projeto. Assim observa-se que o uso de tais diagramas se torna ainda mais importante quando sistemas cientes de contexto são desenvolvidos, dada a necessidade de especificar contexto nesses sistemas.

Nesse âmbito, foi buscado trabalhar com modelagem do contexto através da linguagem UML, uma vez que este padrão de modelagem já é familiar aos desenvolvedores de software. A disseminação da UML ocorre por causa da grande expressividade, abrangendo todas as visões necessárias ao desenvolvimento e implantação de sistemas (BOOCH et al., 2000). A UML tem um grande escopo de uso, podendo ser usada em um conjunto diversificado de domínio de aplicações. No entanto, a incapacidade da UML em representar os conceitos de sistemas cientes de contexto torna particularmente desafiante para os desenvolvedores projetar e comunicar decisões decorrentes do desenvolvimento (FARIAS et al., 2009). Portanto uma extensão da UML para modelagem de contexto visa permitir a representação das principais informações relacionadas com as entidades envolvidas, potencializando uma melhor comunicação das decisões de projeto de sistemas cientes de contexto.

Devido as necessidades apresentadas de compreensão do contexto, a popularidade da UML e as suas limitações apresentadas, o foco desse trabalho é na modelagem de contexto para sistemas cientes de contexto, utilizando a linguagem UML. Como vamos mostrar mais a frente neste trabalho a UML é imprecisa para representar o conceito de contexto se propõe uma extensão da mesma para comportar essa modelagem. Para apoiar a linguagem propõe-se um ambiente de modelagem específico de domínio para tornar possível a modelagem de sistemas cientes de contexto utilizando a linguagem proposta. A ferramenta foi implementada como um plug-in da plataforma Eclipse, que permite representar o conceito de contexto e avaliar a corretude dos modelos criados. Avaliações da extensão e da ferramenta proposta foi realizada através de estudo das respostas dos desenvolvedores perante dois modelos de modelagem, um com a UML pura e outra com a ferramenta desenvolvida.

1.1 Contextualização do Problema

Considerando o âmbito em que se aplica a modelagem proposta, pode-se citar alguns dados que motivam esta pesquisa. De acordo com Satyanarayanan (2010) "A informação ao seu alcance em qualquer lugar, a qualquer hora, tem dirigido a visão para a computação móvel" e também "o acesso ubíquo é uma realidade que hoje é vivida por milhões de usuários em todo mundo", e vemos que a cada dia aumenta as necessidades do ser humano em termos de tecnologia móvel e ubíqua. Sendo assim cada vez mais é necessário desenvolver aplicativos que suportem estas caraterísticas de mobilidade, ubiquidade e adaptabilidade.

Para os desenvolvedores de aplicações móveis e ubíquas o contexto envolve informações que estão relacionadas com uma infinidade de aspectos, tais como os usuários, o ambiente, a temporalidade, locais, etc. Por isso conhecer esses aspectos em detalhes é crucial para proporcionar a flexibilidade e adaptabilidade que os sistemas precisam, a fim de adaptar-se às mudanças nas condições e ambientes dinâmicos (SERRAL; VALDERAS; PELECHANO, 2010). Sendo assim é importante a compreensão do contexto e a modelagem do mesmo para auxiliar os desenvolvedores nesse trabalho.

Para modelagem de aplicações é geralmente utilizada a linguagem UML, por ser uma linguagem muito expressiva, abrangendo as visões necessárias ao desenvolvimento (BOOCH et al., 2000; FOWLER; SCOTTS 2000). No entanto podemos ver que os desafios dos paradigmas impostos pelos sistemas ubíquos, como a interpretação do contexto e a atuação do usuário dentro deste não são hoje totalmente capazes de ser expressadas na UML atual. A heterogeneidade dos elementos que se relacionam em um contexto é incapaz de serem expressados na UML o que dificulta para os desenvolvedores especificar objetivamente as relações entre os elementos de um contexto e consequentemente transportar essas relações para o código.

Um contexto é composto por muitas outras entidades e é bem representado por uma metaclasses da UML onde o mesmo pode ser composto por outras metaclasses, outras entidades que compõe o contexto. A definição de Classe fornecida pela especificação UML não pode ser usada nesse caso para definir contexto pois a especificação determina que a classe é um tipo de classificador cujas características são atributos e as operações e comportamentos de uma classe podem ser chamadas em um objeto (*UML, 2014*). Contexto não tem comportamento e não define características comportamentais que podem ser chamados por outros elementos. Nesse caso o Contexto é uma composição de outras entidades. Então a metaclasse Contexto é introduzida para definir um classificador cuja composição se dá por outras metaclasses e que não têm contrapartidas semelhantes no metamodelo UML.

Foram estudados trabalhos de modelagem que estendem a UML para diferentes objetivos como Sun; France; Ry, (2013), Benselim e Seridi-Bouchelaghem (2013), Sindico; Grassi (2009), Silva; Choren; Lucena (2008). Porém, os modelos pesquisados são limitados na questão de elementos de trabalho e os tipos de aplicações que podem se beneficiar das técnicas elaboradas também quanto a extensão da UML para modelar contexto para o desenvolvimento de sistemas cientes de contexto. E não associam a facilidade de uma ferramenta para apoio junto com a extensão proposta.

Chega-se a conclusão que o problema principal é que a UML é imprecisa para representar o conceito de contexto e este conceito é um dos principais pontos para o desenvolvimento de sistemas cientes de contexto. E como o contexto pode ser obtido numa grande variedade de formas, são necessários meios eficientes e eficazes de modelagem da informação. Um dos maiores problemas nas soluções existentes é a variedade de modelos de

contexto utilizados, bem como as diferentes maneiras de encontrar e acessar as fontes de contexto (NADOVEZA; KIRITSIS, 2014). Outro problema é que as ferramentas acadêmicas e comerciais de modelagem atuais não dão suporte à representação e à validação de modelos de contexto eficientes para auxiliar os desenvolvedores no desenvolvimento de sistema cientes de contexto.

1.2 Questão de Pesquisa

Neste trabalho busca-se auxiliar os desenvolvedores na construção de sistemas cientes de contexto. Procurando resolver o problema com uma linguagem de modelagem usando UML. Com a incapacidade da UML em representar os conceitos de sistemas cientes de contexto necessita-se que se use seus mecanismos de extensão. Entre outros problemas apresentados, é possível destacar, por exemplo, que o metamodelo da UML não oferece suporte à modelagem de elementos importantes de contexto como usuário, dispositivo, localização, atividade etc. Procuramos então a melhor forma de modelar contexto para sistemas cientes de contexto.

Sendo assim a questão chave desta proposta é *"como é possível estender a UML criando um metamodelo que possibilite a modelagem de contexto, para que possa ser utilizada pelos desenvolvedores ao trabalharem em sistemas cientes de contexto?"*

Trabalhou-se na hipótese de estabelecer uma linguagem de modelagem estendendo a UML para comportar os conceitos necessários e uma ferramenta onde a linguagem possa ser utilizada para modelagem de contexto. Para abranger melhor os conceitos de contexto fez-se utilização de ontologias para especificar os elementos de composição do contexto, visando contemplar os principais elementos empregados. Assim, foi definido, um vocabulário comum para compartilhar informações de contexto em um domínio de sistemas cientes de contexto.

1.3 Objetivos

O objetivo principal deste trabalho é desenvolver uma linguagem para modelagem de contexto chamada UML2Context, utilizando-se de metamodelos e estendendo a UML para que suporte todos os elementos necessários para utilização em aplicações cientes de contexto. A modelagem abrange informações dos elementos de contexto, levantados nas análises ontológicas, que são: usuário, localização, tempo, dispositivo, atividade e local. Serão fornecidos modelos para entendimento das suas relações e posteriormente utilização desse modelo em uma ferramenta que apoie a criação dos diagramas para o desenvolvimento.

Como resultado busca-se que a linguagem possa facilitar o entendimento e a tomada de decisões para os desenvolvedores auxiliando na construção de aplicações cientes de contexto, trazendo mais qualidade para os softwares nessa área. Buscou-se também que ferramenta que possa ser usada com a linguagem proposta facilitando o trabalho de modelagem.

1.4 Estrutura do Texto

Este trabalho se divide em oito capítulos. No capítulo dois apresenta-se os diversos assuntos necessários para o entendimento do projeto constituindo a fundamentação teórica do trabalho, bem como uma revisão bibliográfica, cobrindo os assuntos de computação ubíqua, computação móvel, estudo do contexto, UML e metamodelos que foram utilizados para

compor o modelo proposto. Neste capítulo também é mostrado um levantamento das ontologias para contexto que foram utilizadas para extrair o vocabulário utilizado na linguagem.

O capítulo três descreve os trabalhos relacionados que também utilizaram-se de extensões UML para modelagem ou procuraram trabalhar o contexto de alguma forma, alguns criaram linguagens de modelagem a partir da UML ou fizeram modelagem de contexto. Seguindo com a comparação dos trabalhos em uma lista de itens e um resumo de cada um deles e suas limitações analisando os mesmos e elucidando a nova proposta de solução.

O capítulo quatro tem como objetivo apresentar a extensão da UML proposta para modelagem de contexto, nomeada de *UML2Context*. Nele é feito o detalhamento do modelo elaborado, definindo os elementos e suas relações. No capítulo cinco detalha-se a ferramenta que foi desenvolvida, as tecnologias que foram utilizadas. Além disso, são detalhados os passos seguidos para a criação da ferramenta.

O sexto capítulo apresenta a avaliação da linguagem efetuada com a ferramenta, a partir da análise de modelos por desenvolvedores experientes. Com suas respostas pode-se observar a eficácia da linguagem desenvolvida. No sétimo e último capítulo, são feitas as considerações finais do trabalho, bem como destacando as principais contribuições e lacunas que podem gerar trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo tem como objetivo descrever os principais conceitos utilizados ao longo deste trabalho, realizando uma revisão resumida dos principais assuntos pesquisados relacionados com o trabalho e que foram exaustivamente estudados. Conceitos de computação móvel e ubíqua, ciência de contexto e UML são revistos, bem como a extensão da UML que foi utilizada para contemplar a modelagem proposta.

Sendo assim as próximas seções (seção 2.1 e 2.2) abordam os conceitos de computação móvel e ubíqua, principais áreas que se utilizam do contexto e onde ele é fundamental, pois sistemas móveis e ubíquos são na maioria das vezes, sistemas cientes de contexto. Assim estes dois Capítulos trazem um relato da investigação na literatura sobre o estado da arte para estes temas. A próxima seção (seção 2.3) conceitua a ciência de contexto, seus requisitos e classificações, tema principal abordado neste trabalho pois a modelagem concentra-se no contexto. Seguindo, a seção 2.4 conceitua-se UML e a criação de metamodelos, que foram utilizados como base do trabalho para a criação da linguagem de modelagem. Na última seção da fundamentação (seção 2.5) mostra como foram selecionados através dos vocabulários das ontologias, os termos utilizados na linguagem de modelagem.

2.1 Computação Móvel

Nas três últimas décadas, segundo Satyanarayanan (2010), a busca incessante pela informação ao fácil alcance e em qualquer lugar e a qualquer hora, tem estimulado inovações na tecnologia sem fio, hardware portátil eficiente em termos de energia e software adaptável. Assim, desenvolveu-se uma nova classe de dispositivos móveis que se relacionam com os serviços de telecomunicações para oferecer conexões com outros sistemas. Consequentemente trazem a necessidade de aumentar consideravelmente o trabalho de desenvolvimento para sistemas móveis, que são particularmente problemáticos devido aos vários dispositivos de hardware em que podem ser utilizados. Estes dispositivos e as várias formas em que eles expandem a concepção de aplicações informatizadas que antes eram de localização fixa, agora em movimento, estão sendo utilizadas em vários ambientes e contextos, que veremos melhor na próxima seção. Isso leva a novos desafios de design para interação humano-computador, como já dizia Dix et al. (2000). Também Lopes et al. (2013) em seu artigo confirmou que esta classe de sistemas computacionais, que reagem ao contexto, abrem perspectivas para o desenvolvimento de aplicações complexas mais ricas e elaboradas. Essas aplicações exploram o dinamismo das infraestruturas computacionais modernas e cada vez mais a mobilidade do usuário.

Com essa evolução da computação móvel, uma série de problemas surgiram pelo uso de equipamentos móveis que estão em constante mudança de ambiente e usuário. Essas questões levantaram a importância de se levar em conta o conceito de ciência de contexto, que começou a ganhar espaço, pois era visto como uma solução promissora para estes problemas de troca de ambiente e usuários (STRANG; LINNHOFF-POPIEN, 2004). Redes móveis, são geralmente caracterizadas por várias restrições, como espaço de armazenamento, largura de banda, a energia da bateria, onde está sendo usada e as flutuações rápidas na disponibilidade desses recursos; o que torna difícil para o software manter a garantia de qualidade de serviço. Por isso, enquanto concepção de redes sem fio, bem como softwares de aplicativos móveis todas estas questões devem ser levadas em consideração (MANNADE; BHANDE, 2013).

Nesta linha, em seu artigo Satyanarayanan (2010) aborda os principais pontos nos dispositivos móveis a serem explorados e problemas a serem sanados para computação móvel

de forma que ela venha a colaborar com os seres humanos. O autor concluiu que é necessário um avanço nas implementações para os dispositivos móveis a fim de que possam ser usados em diferentes contextos. Em todos os casos softwares integrados em rede para acesso a esses dispositivos com segurança e capazes de entender e ajustar os dados captados para a necessidade das pessoas, serão necessários.

Sendo assim, para aumentar a eficiência e efetividade dessas aplicações, elas terão de ser informadas sobre o contexto em que estão sendo usados a fim de se adaptar automaticamente a ele (NADOVEZA; KIRITSIS, 2014). Cada um dos diferentes ambientes onde o dispositivo móvel foi usado representa uma parte do espaço no qual os sistemas móveis devem atuar levando em consideração os recursos de infraestrutura, sistema, domínio e ambiente. Todos estes itens sugerem que os desenvolvedores devem lidar com a concretização de sistemas interativos móveis inseridos no contexto em que serão utilizados e por quem serão utilizados (DIX et al., 2000).

Essa mobilidade necessária nos aplicativos pode ser tanto física, relacionadas com equipamentos ou usuários, quanto lógica, relacionado ao código e dados (COSTA; YAMIM; GEYER, 2008). Na pesquisa efetuada por Abowd et al. (2005) foi essencial que o software fosse adequado para o uso em ambiente social e físico dos participantes já que os "*paratypes*", características individuais, são adquiridas basicamente por influência do meio, no dia a dia das pessoas. Fica evidente a importância do contexto do usuário no uso dos dispositivos móveis e a importância de softwares que possam captar os dados essenciais em cada contexto. Todos esses conceitos apresentam uma grande quantidade de novos requisitos, metodologias de desenvolvimento e técnicas de modelagem especificamente adaptados para o desenvolvimento de sistemas móveis e também utilizados em parte nos sistemas ubíquos que será abordado a seguir.

2.2 Computação Ubíqua

A computação ubíqua surgiu como um conceito de computação onipresente. Uma das primeiras definições e a mais usada, que define de forma simples e clara a computação ubíqua dada por Weiser (1991) diz que "As tecnologias mais profundas são aquelas que desaparecem. Elas se integram a vida cotidiana até se tornarem indistinguíveis da mesma.". Nesse conceito a computação deve extrapolar os conceitos de computação limitada a um ambiente que é preparado para funcionar em um computador. A computação deve atravessar a fronteira dos centros computacionais e estar no dia a dia das pessoas auxiliando-as nas tarefas. Pois a função da Computação Ubíqua é integrar a tecnologia computacional ao mundo real onde a potência do conceito não vem de quaisquer dispositivos de hardware, mas a partir da interação de deles, tornando os indivíduos mais conscientes das pessoas com as quais estão interagindo sobre as outras extremidades de seus computadores (WEISER, 1991).

Nas aplicações ubíquas é esperado que o acesso do usuário ao ambiente computacional, com um bom desempenho, ocorra em todos os lugares e momentos, por meio de qualquer dispositivo (COSTA; YAMIM; GEYER, 2008). Surge então a dificuldade nos aplicativos que necessitam continuamente se adaptar ao meio ambiente ou a novos dispositivos que são trocados a todo momento (COSTA; YAMIM; GEYER, 2008). De acordo com Abowd et al. (2005) projetar com foco no usuário é um desafio primário em computação ubíqua, pois os paradigmas e características de integração ainda não são familiares. Para desenvolver os projetos e vencer os desafios de avaliação dessas aplicações, estão sendo refinadas técnicas de projeto e prototipagem mais apropriadas para o desenvolvimento de aplicações de computação móvel e ubíqua, que são: protótipos compostos e "*paratypes*"

(características individuais adquiridas basicamente por influência do meio). Para o desenvolvimento de sistemas ubíquos, Weiser (1991) levanta duas questões: localização e escala. Pois, se o computador sabe onde ele se encontra pode adaptar o seu comportamento de forma significativa e computadores em diferentes tamanhos podem se adequar a uma determinada tarefa.

Conforme Costa, Yamim e Geyer (2008), as principais questões e desafios da computação ubíqua são: heterogeneidade, escalabilidade, confiança e segurança, privacidade, espontaneidade e interoperabilidade, mobilidade, consciência de contexto, sensibilidade ao contexto, gerenciamento de contexto, interação do usuário transparente e invisibilidade. Considera-se também um desafio a elevada distribuição, heterogeneidade, dinamismo e mobilidade dos ambientes ubíquos (LOPES et al., 2013). Os desenvolvedores então precisam lidar com todas estas questões ao desenvolver sistemas ubíquos com qualidade.

Vemos então que o objetivo dos novos aplicativos a serem desenvolvidos são a mobilidade e ubiquidade necessitando assim de uma abordagem reativa ao acesso das informações de contexto e que escondam a complexidade do ambiente, no entanto temos um número limitado de linguagens e ferramentas disponíveis para isso (COSTA; YAMIM; GEYER, 2008). E trabalhar com o contexto independente da linguagem requer primeiramente conhecimento sobre a ciência de contexto que será incorporada na aplicação.

2.3 Ciência de Contexto

Uma das principais questões de pesquisa na área de computação ubíqua é a ciência de contexto, no que se refere à capacidade das aplicações para realizar mudanças nas características do ambiente ubíquo que interessam para a mesma e responder a essas mudanças através de um processo de adaptação (LOPES et al., 2013). A definição de contexto vem evoluindo nos últimos anos paralelamente ao desenvolvimento de sistemas cientes de contexto. Em Schmidt, Beigl e Gellersen (1999) os autores debateram sobre o contexto simples e o contexto categorizado, onde o desafio foi identificar os dados relevantes para capturar o contexto coerente para utilização nas aplicações. Nessa análise as categorias foram divididas em:

- *Fatores humanos:* essa categoria é subdividida em usuário (hábitos, estado emocional, fisiológico, etc), ambiente social (outras pessoas envolvidas, relacionamento social, dinâmica de grupo, etc) e tarefas (atividades, tarefas, objetivos gerais etc).
- *Ambiente físico:* essa categoria é subdividida em localização (posição absoluta, posição relativa, co-localização), infraestrutura (recursos ao redor e interações de comunicação no ambiente) e condições físicas (ruído de fundo, nível de luz do ambiente, pressão, etc.).

No entanto a definição do que é o contexto e o que ele engloba, pensando em sistemas computação, é um assunto que nota-se ser muito abordado entre os pesquisadores. Conforme o Foldoc (2014) contexto é “aquilo que envolve e dá sentido à outra coisa”. Dey e Abowd et al. (2000) disseram então que uma aplicação é dita sensível ao contexto quando utiliza informações de contexto a fim de fornecer serviço ou informação relevantes ao usuário de acordo com a ação que vai executar. E forneceram uma definição muito referenciada de contexto:

> "O contexto deve ser qualquer informação que pode ser utilizado para caracterizar a situação de uma entidade, em que uma entidade pode ser uma pessoa, um local, um objeto físico ou computacional." (DEY: ABOWD, 2000).

Posteriormente Dey (2001) mais uma vez confirma, dizendo que o contexto é tudo sobre toda a situação relevante para a aplicação e seu conjunto de usuários. Sendo assim um sistema é ciente de contexto se esse sistema utiliza os dados coletadas do contexto em que está inserido para fornecer informações ou serviços relevantes para o usuário da aplicação. E a relevância das informações disponíveis depende da tarefa que o usuário terá que executar.

Em Dix et al. (2000) é definido uma taxonomia para contexto baseado nas interações humano-computador. Dix et al. (2000) defende que um dispositivo móvel opera em um contexto muito amplo, incluindo infraestrutura de rede, computacional, sistema, domínio da aplicação e ambiente físico. E conforme Costa, Yamim e Geyer (2008) a pesar do conceito de contexto ser mais claro na computação ubíqua a computação móvel também introduziu a ideia de ciência de contexto, usando o contexto para fornecer informações ou serviços aos usuários.

Desse modo, observa-se que a utilização do contexto diverge em alguns pontos entre os autores e o tema de ciência de contexto é uma das principais questões na computação ubíqua. Em sistemas cientes de contexto o contexto vai impactar diretamente no comportamento do sistema para facilitando a interação sistema-usuário. Sendo importante para qualquer abordagem de modelagem de contexto levar em conta os seguintes requisitos (STRANG; LINNHOFF-POPIEN, 2004):

a) *Composição distribuída* (DC): composição e a administração de um modelo de contexto e seus dados vai variar de acordo com o tempo, topologia de rede e fonte;

b) *Validação parcial* (PV): ser capaz de validar parcialmente o conhecimento contextual;

c) *Riqueza e qualidade de informação* (QUA): avaliar a qualidade de uma informação fornecida por um sensor, pois a mesma varia ao longo do tempo;

d) *Incompletude e ambiguidade* (INC): deve ter capacidade de tratar o conjunto de informações contextuais disponíveis de entidade e ambiente que podem estar incompletos ou ambíguos;

e) *Nível de formalidade* (FOR): necessidade de descrever fatos contextuais e inter-relações de uma maneira precisa e rastreável;

f) *Aplicabilidade aos ambientes existentes* (APP): o modelo de contexto deve ser aplicado à infraestrutura de ambientes de computação cientes de contexto.

Em termos de recursos, há três categorias que uma aplicação ciente de contexto deve suportar (DEY, 2001; BULCÃO, 2006):

a) *Apresentação de informações e serviços para o usuário*: a habilidade de diferenciar entre informações de contexto e serviço;

b) *Execução automática de um serviço:* habilidade de executar um serviço automaticamente de acordo com as informações atuais do usuário e do contexto;

c) *Marcação das informações de contexto para apoiar na posterior recuperação de informações:* habilidade de unir as informações de contexto que descrevem a situação do usuário e que possam ser acessadas posteriormente.

O objetivo das pesquisas sobre conceito de contexto visa desenvolver modelos de contexto uniformes. Os autores Strang e Linnhoff-Popien, C. (2004), classificaram várias abordagens desse conceito:

a) *Modelos de Chave-Valor*: estrutura de dados mais simples para a modelagem de informações contextuais que utiliza pares de valores-chave para modelar o contexto. Frequentemente é usada em estruturas de serviço distribuídos e os serviços são descritos como uma lista de atributos simples em forma de valores-chave;

b) *Esquema de Marcação de Modelos*: é uma estrutura de dados hierárquica que consiste em tags de marcação com atributos e conteúdo que são geralmente usados na abordagem de perfis. Costumam basear-se sobre a serialização de um derivado do padrão genérico "*Markup Language*";

c) *Modelos Gráficos*: utiliza a UML com seus diagramas como instrumento de modelagem UML, que pela sua estrutura genérica, pode ser adaptado para modelar o contexto. Esta abordagem é particularmente usada para derivar um modelo ER;

d) *Modelos Orientados a Objetos*: com o objetivo de empregar os principais benefícios de qualquer abordagem orientada a objetos, encapsulamento e reutilização, ele cobre parte dos problemas decorrentes das dinâmicas do contexto em ambientes Ubíquos. O processamento de contexto é encapsulado ao nível de objeto;

e) *Modelos Baseados em Lógica*: com um alto grau de formalidade um conjunto de regras de um sistema formal são aplicadas em um modelo de contexto baseado lógica, onde o contexto é definido como fatos, expressões e regras. A informação contextual é adicionada, atualizada ou excluída do sistema baseado em termos de fatos ou a partir das regras do sistema;

f) *Modelos de Base Ontológica*: ontologias são utilizadas para especificar conceitos e inter-relações. Foram propostos modelos de contexto baseado em ontologias devido aos seus pontos fortes em normalização e formalidade. Fornece uma maneira uniforme para a especificação de conceitos fundamentais do modelo, assim como sub-conceitos e fatos permitindo o compartilhamento do conhecimento do contexto e reutilizando-o em um sistema de computação ubíqua.

A Tabela 1 compara as abordagens explicadas e sua associação com os requisitos levantados, mostrando a aderência de cada abordagem a cada requisito. Os requisitos marcados com sinal de mais, são os que mais fortemente são comtemplados em cada abordagem.

Dos cenários analisados utilizamos a abordagem de Modelos Gráficos que colocamos em negrito na Tabela 1, pois modelos gráficos utilizam a UML para modelagem de softwares utilizando seus conceitos e suas técnicas de extensão. Os modelos gráficos são amplamente conhecidos e utilizados em ambiente comercial, pela sua rápida aplicação e fácil entendimento por parte da equipe de desenvolvimento. Pode-se ver nesta tabela que a abordagem destaca os pontos onde o modelo gráfico é mais atuante que são: qualidade, formalidade da linguagem e aplicabilidade. Também para padronizar o vocabulário

trabalhado no modelo de linguagem utilizamos as ontologias, onde encontramos muitos trabalhos sobre a padronização dos termos para o contexto.

Tabela 1: Tabela de comparação entre os trabalhos relacionados.

Abordagem/Requisito	DC	PV	QUA	INC	FOR	APP
Chave-Valor	-	-				+
Marcação de Modelos	+	++	-	-	+	++
Modelo Gráfico		**-**	**+**	**-**	**+**	**+**
Orientado a Objetos	++	+	+	+	+	+
Baseado em Lógica	++	-	-	-	++	
Base Ontológica	++	++	+	+	++	+

Fonte: Traduzido livremente de Strang e Linnhoff-Popien, (2004).

Apoiado nos modelos gráficos e ontológicos o foco do trabalho é a modelagem para auxiliar no desenvolvimento de aplicações cientes de contexto. Onde o principal ponto que abordaremos nesse trabalho é a modelagem de contexto, para auxiliar no desenvolvimento de softwares cientes de contexto onde percebendo o estado do usuário e do ambiente, devem ser colhidas informações de contexto ao aplicativo (COSTA; YAMIM; GEYER, 2008). Assim para que possamos trabalhar com o modelo gráfico, precisamos de uma linguagem familiar e que possa ser adaptada para as necessidades visuais existentes, portanto e pelas suas muitas facilidades trabalharemos com a UML e sua abordagem de metamodelos.

2.4 UML e Metamodelo

A modelagem é uma parte essencial das atividades de entendimento e construção de um software para que ele atenda aos requisitos. Por isso, modelos são criados e mantidos para entender a estrutura e o comportamento desejado, visualizar e controlar a arquitetura e compreender melhor o sistema. Projetos de software malsucedidos falham em relação aos aspectos únicos e específicos em cada situação, mas projetos bem-sucedidos se assemelham em alguns aspectos e um dos principais é a utilização da modelagem (BOOCH et al., 2000).

Modelos UML têm sido amplamente adotados tanto na academia quanto na indústria durante todo o ciclo de vida de um software. Fowler e Scotts (2000) argumentam que “A UML é a sucessora da onda de métodos de análise e projeto orientados a objetos (OOA & D) que surgiu no final dos anos oitenta e no início dos anos noventa. Mais especificamente, ela unifica os métodos de Booch Rumbaugh (OMT) e Jacbson.”. A UML que tornou-se um padrão OMG (*Object Management Group*) após passar por um processo de padronização pela organização é utilizada amplamente até hoje, seguindo o paradigma de orientação a objetos (FOWLER; SCOTTS, 2000).

A especificação UML possui uma abordagem familiar padrão que modela uma classe usando três itens: nome da classe, atributos da classe e métodos da classe. Nesse caso a classe é um tipo de classificador cujas características são atributos e as operações desta classe podem ser chamadas em um objeto. Possui um amplo alcance e deve ser estruturada de forma modular, permitindo selecionar apenas as partes da linguagem que lhe interessa (UML, 2014). No entanto esses padrões da UML podem ser alterados de acordo com a necessidade seguindo um critério.

A linguagem UML fornece ferramentas para análise, projeto e implementação de softwares assim como modelagem de negócios e processos (UML, 2014). Sua estrutura é

definida em nível abstrato e é estendida usando princípios de orientação a objetos para definir uma superestrutura UML (UMLBase, 2014).

A UML (2014) define:

- Notação: é a parte gráfica, a sintaxe da linguagem de modelagem definindo como os itens e conceitos serão modelados;
- Metamodelo: diagrama que define a notação, geralmente um diagrama de classe.

Conforme Booch et al. (2000) embora a UML ofereça uma linguagem padrão para preparação de projetos de software, nenhuma linguagem poderá ser suficiente para expressar todas as nuances de todos os modelos em todos os domínios o tempo todo. Assim a UML foi destinada a ser aberta tornando possível estendê-la de forma controlada através dos seguintes mecanismos de extensibilidade:

- Estereótipos: estende o vocabulário UML, permitindo a criação de novos tipos de blocos derivados dos já existentes, mas específicos ao seu problema;
- Valores atribuídos: estende as propriedades de um bloco de construção da UML, permitindo a criação de novas informações na especificação desse elemento;
- Restrições: estende a semântica de um bloco de construção da UML, permitindo adicionar novas regras ou modificar as existentes.

Quando as notações se limitam para o nosso uso e se tem a necessidade de modelar conceitos não previstos no modelo normal, a UML lhe dá a oportunidade de estender o seu metamodelo. A típica função de um metamodelo é definir a semântica de como os elementos se instanciam em um modelo (UML, 2014). O metamodelo da UML é composto pelo que se pode chamar de pacotes logicamente separados em camadas. Conforme UML (2014) e UMLBase (2014) seus princípios de projeto e arquitetura são:

- Modularidade: princípio de forte coesão e fraco acoplamento, capacidade de separar em módulos físicos os objetos, as classes e as funcionalidades, organiza recursos no que chama de metaclasses. E possui também privacidade de informações.
- Camadas: são aplicadas de duas maneiras para o metamodelo UML. Primeiro a estrutura do pacote é disposta em camadas para separar as construções fundamentais da metalinguagem das construções de nível mais elevado. Segundo, um padrão de arquitetura de 4 camadas metamodelo é consistentemente aplicado. O metamodelo UML é dividido em um modelo de camadas no padrão TIER IV[4]: instâncias de nível de usuário das classes do modelo, o modelo, o metamodelo, o meta-metamodelo. Cada nível contém um grupo de elementos que não se misturam com elementos de outras camadas. (Obs.: O TIER é uma classificação para qualidade de infraestruturas. Chamado de "Tier" ("camada" em inglês), o padrão mundial de classificação foi criado especialmente para Data Centers pelo consórcio Uptime Institute e validado pelo Owner Advisory Committee e é aceito em mais de 40 países.).
- Particionamento: usado para organizar áreas conceituais dentro da mesma camada. No metamodelo UML a divisão é mais rústica, chamada de Particionamento Horizontal. Como por exemplo particionar em pacotes os diferentes níveis de elementos. Aumenta a coesão e reduz o acoplamento entre pacotes.

[4] Padrão TIER IV: http://uptimeinstitute.com/TierCertification/

- Extensibilidade: o modelo UML tem a possibilidade de ser estendido para comtemplar d diferentes modelos e linguagens que pode ser feito de duas maneiras:
    a. Novo dialeto UML: pode-se usar perfis para personalizar a linguagem para plataformas específicas definindo novo dialeto da UML como: J2EE, EJB, .NET e domínios como financeiro, marketing, científico;
    b. Nova linguagem UML: reutilizando parte do pacote InfrastructureLibrary e aumentando o mesmo com metaclasses e metarelationships apropriadas pode ser especificada uma nova linguagem;
- Reutilização: na definição do metamodelo UML é feita reutilização de construções pré-definidas na biblioteca metamodelo flexível (InfrastructureLibrary), bem como outras arquiteturas relacionadas tal como o Meta Object Facility (MOF) e o Common Warehouse Metamodel (CWM).

Os metamodelos permitem que a UML possa ser adaptada para diferentes utilizações englobando outros conceitos não previstos inicialmente na mesma.

2.4.1 As Camadas da Hierarquia do Metamodelo

Para definir os metamodelos UML quatro camadas são estabelecidas, os modelos de extensão propostos derivam a partir dessas camadas. São elas (UMLBase, 2014):

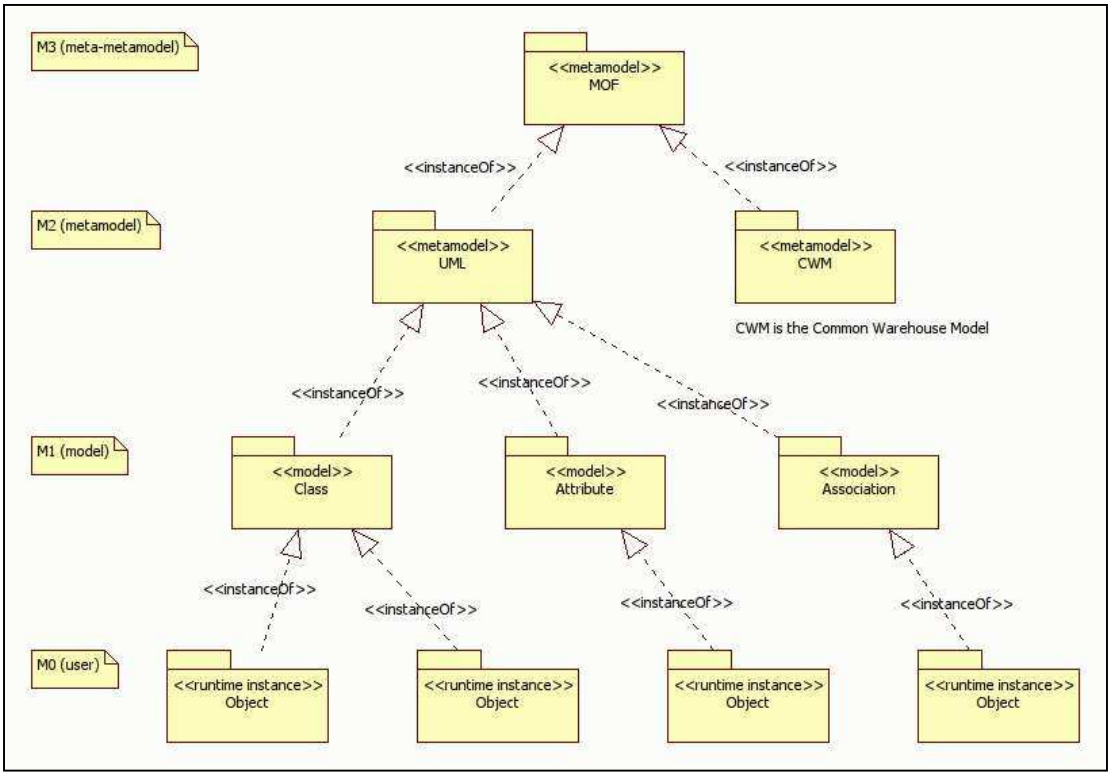
- Camada M3 (meta-metamodelo): principal responsabilidade desta camada é definir a linguagem para especificar um metamodelo; É normalmente mais compacto do que o metamodelo que ele descreve e pode definir vários metamodelos, mas é desejável que compartilhem filosofias de design comuns e construções. Exemplo: MOF;
- Camada M2 (metamodelo): principal responsabilidade desta camada é definir uma linguagem para especificar modelos. E é uma instância de um meta-metamodelo, assim todos os elementos do metamodelo são uma instância de um elemento no meta-metamodelo. Exemplo: UML e o OMG Common Warehouse Metamodel (CWM);
- Camada M1 (modelo): principal responsabilidade desta camada é definir linguagens que descrevem domínios semânticos, que permitem que os usuários modelem uma grande variedade de diferentes domínios de problemas. E é uma instância de um metamodelo. Exemplo: modelo de usuário;
- Camada M0: uma instância de nível de usuário das classes M1, contém as instâncias de tempo de execução de elementos definidos em um modelo.

Na Figura 1 é possível observar as camadas do diagrama UML mostrando suas divisões onde os elementos são agrupados de acordo com suas características. Pode-se ver os objetos que são utilizados pelas classes que estão separados nas camadas. Nesse exemplo a classe faz parte de um modelo que está em outra camada e que são instanciados na camada meta-metamodelo MOF.

Para construção do nosso modelo, foi utilizada a abordagem de Modelo Gráfico visto na seção 2.3, e que foi utilizada na linguagem UML para modelar o contexto. O ponto forte deste modelo está definitivamente no nível da estrutura. Eles são usados principalmente para descrever a estrutura do conhecimento contextual e obter um código (BAUER, 2003) ou um modelo ER (HENRICKSEN, INDULSKA, 2004), o que é muito importante no sentido da exigência de aplicabilidade (STRANG; LINNHOFF-POPIEN, 2004).

Neste trabalho abordaremos análises que utilizaram os conceitos de extensão de UML e metamodelo para efetuar modelagens com focos diferenciados. Mas para cada pesquisa a extensão da UML foi aplicada de maneira formal e descritiva introduzindo os conceitos em metamodelos iniciais que são detalhados e aprofundados no decorrer dos estudos.

Figura 1: As camadas de um diagrama UML Metamodelo.



Fonte: UMLBase (2014).

2.5 Definições de Termos para Modelagem

A informação do contexto pode ser obtida numa grande variedade de formas, sendo assim, são necessários meios eficientes e eficazes de modelagem da informação (NADOVEZA; KIRITSIS 2014). Para auxiliar na modelagem de sistemas cientes de contexto primeiramente é necessário identificar os principais termos utilizados para definir contexto. É importante destacar que sistemas cientes de contexto utilizam mecanismos para extrair informações e assim adaptar o aplicativo ao contexto para que o usuário possa fazer um melhor uso do software. Considerando as origens da informação de contexto, ele pode ser recuperado a partir de fontes internas, dados e informações que são coletados pelo sistema,

como o estado do sistema e eventos ocorridos ou informações externas, como dispositivo, localização e outros (NADOVEZA; KIRITSIS 2014).

Ainda sobre a classificação do contexto, alguns autores (NADOVEZA; KIRITSIS 2014) tem classificado o mesmo dependendo da forma como os dados são capturados em três grupos: (1) *contexto físico* que é representado com dados provenientes de sensores físicos, tais como aparelhos de GPS, sensores de luz; (2) *contexto virtual* que é baseado em informações provenientes de aplicações ou serviços de software; e (3) *contexto lógico* que é derivado de informações provenientes de diversas fontes de informação, combinando informações a partir de sensores físicos e virtuais.

Além disso a literatura atual tem procurado em ontologias os vocabulários de contexto para definição dos principais termos que são utilizados. Schmidt, Beigl e Gellersen (1999) abordam algumas categorias importantes como, por exemplo, fatores humanos, usuário e ambiente físico. No entanto, Dey (2001) especifica que o contexto é toda a informação que caracteriza uma entidade, nesse caso informações humanas e físicas. É possível extrair diversas informações destas duas categorias, e sua utilização vai depender da relevância para interação do usuário com aplicativo. Sendo assim um dos maiores problemas nas soluções existentes é a variedade de modelos de contexto utilizados, bem como as diferentes maneiras de encontrar e acessar as fontes de contexto (NADOVEZA; KIRITSIS 2014). Todo sistema e estrutura usa seu próprio formato para descrever contexto e para montar um metamodelo que auxilie na modelagem de sistemas é preciso ter uma definição integrada e consistente. Isso permitirá que as informações dos principais conceitos de contexto possam ser representadas de forma consistente.

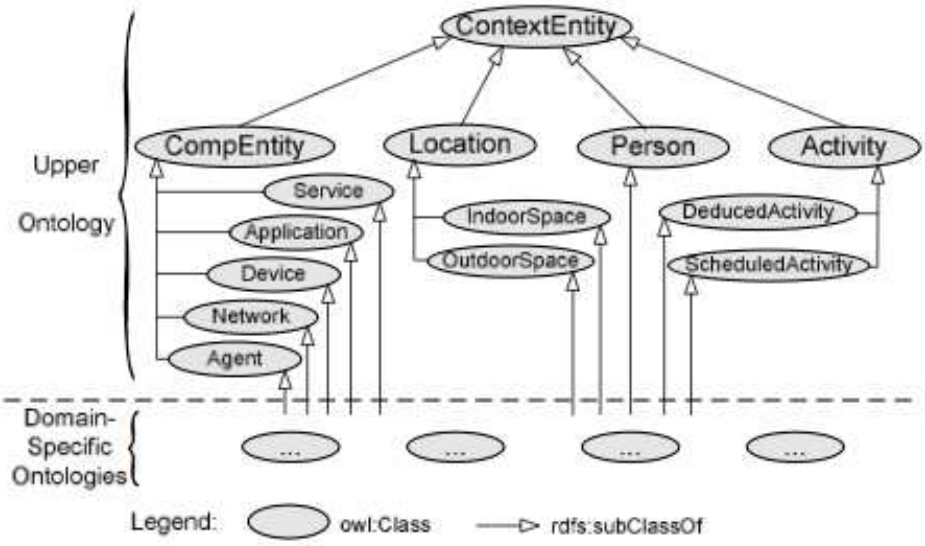
Para efetuar uma modelagem incluindo os conceitos principais de contexto é preciso definir os itens que serão considerados. Para isso utilizou-se a base ontológica em alto nível para guiar a montagem do metamodelo, pois as ontologias são um instrumento promissor para especificar conceitos e inter-relações (STRANG; LINNHOFF-POPIEN, 2004). Ontologia de alto nível ou infraestrutura ontologia superior é definida como um conjunto genérico de classes e as relações de propriedades que podem ser usadas em diferentes aplicações e domínios (NADOVEZA; KIRITSIS 2014). Na análise de Strang e Linnhoff-Popien a abordagem cobriu muito bem as exigências de requisitos para modelagem de contexto. Como o objetivo não é aprofundar a pesquisa em ontologias, pois já existem diversos trabalhados detalhando ontologias de contexto, serão utilizados estes trabalhos existentes para a definição do vocabulário e a partir dele estabelecer uma sintaxe abstrata que terá os principais conceitos e seus relacionamentos.

No trabalho de Gu et al. (2004) e Bulcão e Pimentel (2005), foram desenvolvidos modelos de contexto baseados em ontologias, os quais fornecem um vocabulário para representação de conhecimento sobre um domínio, nesse caso o contexto. Ontologia de contexto define um vocabulário comum para compartilhar informações de contexto em um domínio computação ubíqua e também móvel, incluindo definições de conceitos básicos do domínio e as relações entre eles. Gu et al. (2004) refere como o entendimento compartilhado de alguns domínios, que geralmente é concebida como um conjunto de entidades, relações, funções, axiomas e instâncias. A ontologia de contexto deve ser capaz de capturar todas as características de informações de um contexto. Além disso, deve representar várias informações de contexto em um ambiente de aplicações cientes de contexto, o que é realmente uma tarefa difícil que muitos pesquisadores enfrentam (GU et al., 2004) e que pode ser sanado com o uso de uma abordagem ontológica.

Gu et al. (2004) e Bulcão (2005) utilizam a linguagem OWL (Ontology Web Language) para definir ontologias, que foi escolhida para definir os modelos de contextos e definir as ontologias de contexto. Esta linguagem foi escolhida pelos autores, porquê é muito expressiva em comparação com outras linguagens de ontologias, possuindo a capacidade de suportar a interoperabilidade semântica para compartilhar conhecimentos entre diferentes sistemas, porquê permite o reconhecimento automatizado para ser usado por processos automatizados e por fim, outras técnicas se fundiram em OWL para se tornar um padrão aberto W3C (World Wide Web Consortium), (GU et al., 2004).

Na Figura 2, o modelo mostra os elementos onde a ontologia superior define o conceito da entidade contexto e os conceitos básicos de pessoa, localização, entidade computacional e atividade. *ContextEntity* fornece um ponto de entrada de referência para declarar a ontologia superior, onde cada instância do *ContextEntity* apresenta um conjunto de conceitos básicos de pessoa, localização, *CompEntity* e atividade, onde seus detalhes são definidos nas ontologias específicas de domínio que podem variar de um domínio para outro, definido todas as classes descendentes das classes básicas em um ambiente de casa inteligente e um conjunto de propriedades e relações que estão associados a essas (GU et al., 2004).

Figura 2: Diagrama de hierarquia para ontologias de contexto superior.

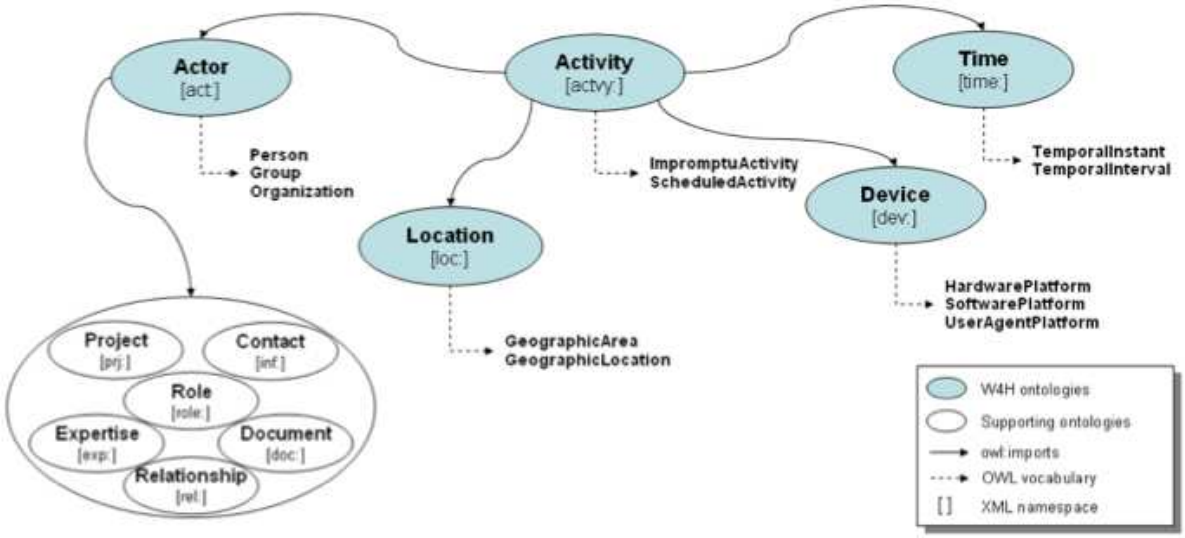


Fonte: Gu et al. (2014).

O modelo na Figura 3 mostra uma visão geral do modelo de contexto ontológico de Bulcão Neto e Pimentel (2005) com base nas dimensões W4H. O modelo de contexto representa os conceitos básicos de atores, localização, tempo, atividades e dispositivos, bem como as relações entre estes conceitos. As cinco dimensões semânticas chamada W4H engloba itens e questões a serem abordadas são identidade (quem), localização (onde), tempo (quando), atividade (o que) e perfis de dispositivos (como). É um modelo de contexto semântico independente de domínio com ontologias de alto nível.

Bulcão (2006) também analisou diferentes modelos ontológicos identificando o vocabulário que cada um deles utilizou (descrito na Tabela 2). São eles: SOUPA, CONON e SeCoM das respectivas iniciativas CoBrA, SOCAN e SCK (infraestruturas), utilizou-se a tabela para evidenciar as informações contextuais relacionadas independente do modelo ontológico.

Figura 3: Modelo de contexto semântico com ontologias de alto nível.



Fonte: Bulcão Neto e Pimentel (2005).

Tabela 2: Comparação entre os modelos ontológicos analisados por Bulcão.

Modelo	Informação contextual
SOUPA	perfil de usuário, localização, tempo, dispositivo, evento, ação, crença, desejo, intenção e privacidade
SeCoM	perfil de usuário, localização, tempo, dispositivo, evento e atividade
CONON	usuário, localização, tempo, dispositivo, rede, aplicação, serviço, agente e atividade

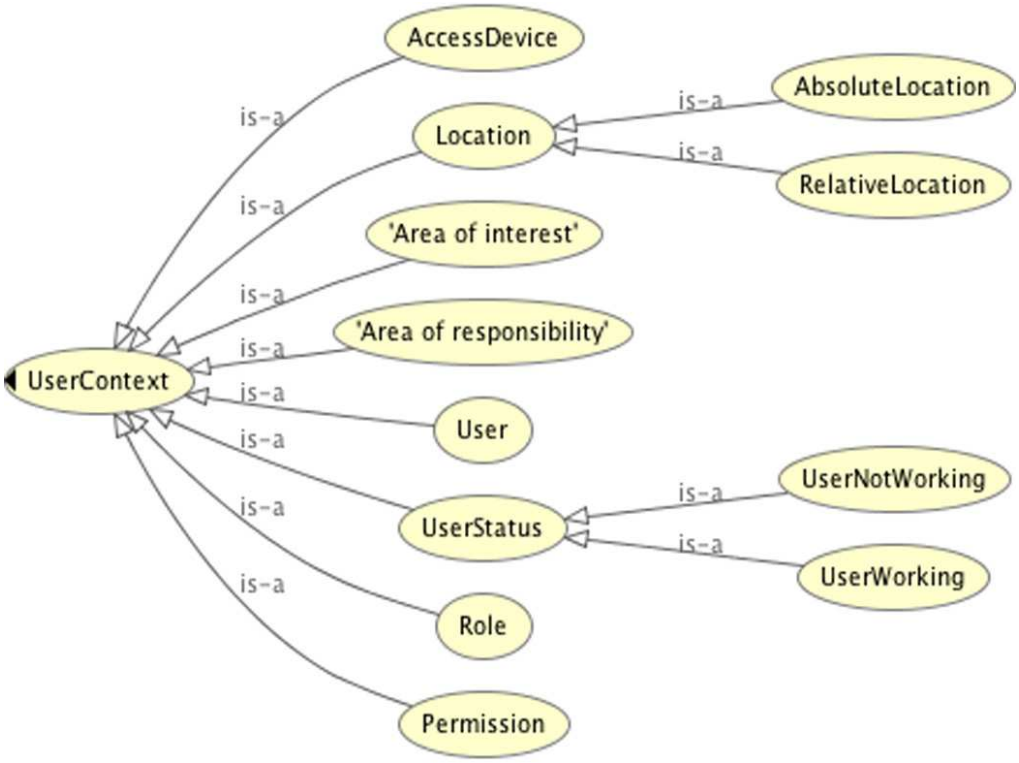
Fonte: Bulcão Neto (2006).

Em Nadoveza e Kiritsis (2014), os autores levantaram separadamente itens de contexto que são: usuário, estado do utilizador, localização, papel, dispositivo, área de responsabilidade, área de interesse e permissões; Além disso, foram definidos itens de contexto empresarial que são: atividade empresarial, estado da atividade, resultado da atividade, recursos de negócio, evento de negócios. Contexto de usuário que é abordado no trabalho neste caso representa as várias entidades e conceitos que são considerados relevantes para a descrição da situação atual do usuário e sua interação com o sistema. Os itens de contexto devem neste caso capturar as informações sobre a identidade do usuário, sua localização física, bem como sua interação com os outros usuários. A Figura 4 ilustra a hierarquia de classes da ontologia de contexto do usuário com os itens selecionados pelos autores.

Na maior parte das ontologias de contexto existentes variáveis de contexto que geram modelos genericamente diferentes que carecem de expressividade e em geral eles se concentram no contexto do usuário que é um dos principais papéis (NADOVEZA; KIRITSIS 2014). Portanto partindo de todas as especificações de conceitos para contexto, foram escolhidos os termos mais relevantes e que possam ser utilizados para modelagem de qualquer sistema que utilize ciência de contexto.

O modelo que foi proposto de contexto está de acordo com os principais pontos levantados nas ontologias analisadas de contexto e em alguns casos chamado de contexto superior, pois poderia englobar outros contextos. O modelo também deve ser capaz de ser utilizado em qualquer situação em que haja contexto. Assim focou-se nos itens que se repetem nos modelos ontológicos analisados e que são genéricos, podendo assim serem utilizados amplamente. Dessa forma, se espera que os principais itens de contexto, estejam incorporados no modelo.

Figura 4: Modelo de classe *UserContext* e suas subclasses.



Fonte: Nadoveza e Kiritsis, 2014.

Temos assim os principais termos a serem modelados e que serão abordados, mas profundamente em nosso modelo:

- Contexto: o contexto em si é visto como um elemento que agrupará outros elementos indispensáveis para o reconhecimento do contexto;
- Usuário: refere-se ao usuário do sistema, o indivíduo interessado nos resultados que serão gerados;
- Dispositivo: refere-se ao equipamento de hardware onde será executado o sistema e com o que o usuário irá interagir;

- Atividade: refere-se à atividade que está sendo feita pelo usuário ou sistema e no qual se está interessado em obter respostas;
- Localização: refere-se ao local físico e geográfico onde ocorre os eventos e que interessa ao usuário e a aplicação;
- Tempo: refere-se ao registro do tempo como datas, horas e duração da tarefa ou atividade.

Os elementos que estão destacados foram selecionados nas ontologias analisadas mencionadas no capítulo, e que foram selecionadas pela sua relevância, abrangência e importância dentro das ontologias. Observamos os elementos se repetiram em diferentes levantamentos de termos ontológicos, buscando englobar os pontos relevantes e imprescindíveis para o contexto. Estes termos foram os selecionados neste trabalho buscando assim englobar, senão todas, mas a maior parte das informações de contexto pertinentes para serem utilizadas por um sistema ciente de contexto. Como as tecnologias são muito dinâmicas deixaremos o modelo aberto a inclusão de mais elementos ou de atributos nos elementos para que o modelo possa ser adaptado à medida que seja necessário.

3 TRABALHOS RELACIONADOS

Diversos trabalhos foram estudados na linha de modelagem para softwares que utilizam os conceitos de ciência de contexto, com o objetivo de identificar as lacunas para a criação de um modelo. Foram escolhidos quatro artigos para análise por apresentarem questões como modelagem de sistemas utilizando a extensão da UML ou abordando a modelagem de contexto junto com a UML. Como não encontramos especificamente artigos que efetuassem modelagem estendendo a UML para modelar contexto, procurou-se artigos que abordassem modelagem de sistemas com UML e que houvesse semelhança com o tema abordado no trabalho. Nessa linha foi encontrado dois artigos que estendem a UML para modelagem de contexto com abordagens diferente da que levantamos, mas muito apropriada. Outro artigo que não estende a UML mas aprofunda a modelagem de contexto que é o que buscamos fazer, e por último, um artigo muito apropriado que não aborda o contexto, mas que desenvolve uma linguagem semelhante a proposta neste trabalho estendendo a UML. Através destes trabalhos foram encontradas oportunidades de contribuição nesta área de modelagem de contexto para sistemas cientes de contexto.

3.1 Análise de Propostas de Trabalhos Relacionados

Nesta seção serão apresentados trabalhos científicos relacionados com a área pesquisada. Alguns trabalhos efetuaram a extensão da UML desenvolvendo um metamodelo, cada qual focando em uma especialidade de sistema. Nesses os metamodelos especificados foram desenvolvidos para posterior uso em uma linguagem de modelagem ou apenas na definição teórica do modelo.

3.1.1 Kermeta

Em Sun, France e Ray (2013) é proposto a decomposição dos conceitos de Ciência de Contexto em modelos de classe UML descrevendo uma abordagem de divisão do modelo de classe que leva em consideração ambas as invariantes estruturais e contratos de operação que são expressas no OCL (Object Constraint Language). Uma invariante é uma asserção a respeito de uma classe, significa que a invariante é adicionada a pré-condições e pós-condições associadas a todas operações de determinadas classes (BOOCH et al., 2000). Utilizando essa abordagem ele decompõe um grande modelo em fragmentos, em que cada fragmento contém elementos do modelo original que são necessárias para analisar o subconjunto das invariantes e dos contratos de operação no modelo proposto. Foi utilizado o modelo de classe UML, com invariantes e contratos de funcionamento que são expressos na OCL. Então para particionar o modelo Sun, France e Ray especificaram dois passos, e a utilização de quatro algoritmos para ajudar a efetuar a divisão e que é descrito resumidamente.

O primeiro passo é criação de um gráfico de dependência. Com um modelo de classe de entrada, levando em consideração as restrições OCL, é feita uma análise e produzido um gráfico de dependência que se relaciona aos seus elementos do modelo e contratos de operação, contendo classes gerais e outras classes referenciadas com suas propriedades. As dependências entre os elementos do modelo são determinadas por relações definidas no metamodelo UML. Então um gráfico de dependência é criado por um algoritmo a partir das dependências computadas e relacionamentos definidos no metamodelo UML. Ele possui nós e arestas, onde cada nó representa um elemento de modelo como classes, atributos, operações e

invariantes e cada aresta representa uma dependência entre dois desses elementos. O primeiro algoritmo usado gera um gráfico de dependência que auxilia na decomposição de um modelo em fragmentos.

O segundo passo faz a decomposição do modelo em fragmentos. Utiliza um gráfico de dependência para gerar os critérios de divisão, que são usados para extrair um ou mais fragmentos do modelo a partir do modelo de classe para serem analisados separadamente. É feito da seguinte forma:

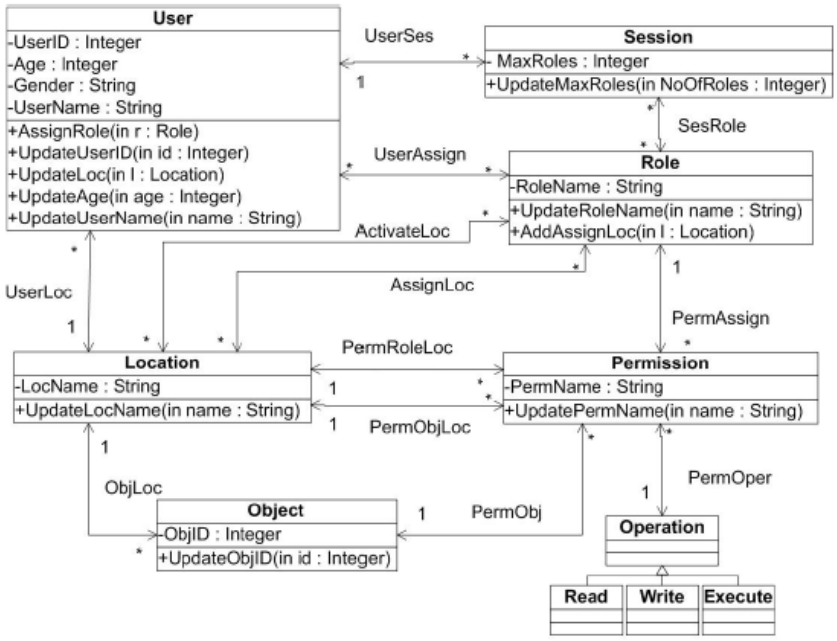
- É identificado os elementos do modelo que não estão envolvidos na análise, nesse caso usa-se um algoritmo (o segundo) identifica elementos irrelevantes no modelo e que são removidos a partir do modelo de classe.
- O próximo passo é identificar os elementos do modelo que estão envolvidos em um problema de análise local, refere-se a uma análise que pode ser realizada dentro do limite de uma classe. Para isso utiliza-se o terceiro algoritmo que identifica elementos do modelo que estão envolvidos em problemas de análise de variáveis locais;
- Nesse ponto é preciso decompor o modelo em fragmentos com o quarto algoritmo. Este algoritmo decompõe um gráfico de dependência em fragmentos do modelo, calculando um conjunto de critérios de corte, onde cada um é composto por um conjunto de operação e vértices de invariantes. Então cada critério de corte é usado para gerar um novo gráfico de dependência que representa um fragmento de modelo.

Para efetuar todo o processo desenvolvido um protótipo de pesquisa com a ferramenta Kermeta (ferramenta "*metamodeling*" orientada a aspectos) para investigar a viabilidade do desenvolvimento de ferramenta de apoio para esta abordagem de divisão. Nesse protótipo é informado um arquivo EMF Ecore (que descreve um modelo de classe de design UML) e um arquivo de texto com os invariantes OCL e especificações de operações. A partir destas informações é produzido uma lista de fragmentos de modelos e o protótipo gera um gráfico de dependência de um modelo de classe UML com invariantes OCL e especificações de operação.

Com os algoritmos descritos, foi procurado acelerar o processo de verificação para grandes modelos. No entanto fica claro que o processo não garante que mais de um fragmento foi produzido a partir de um modelo para ser analisado de forma independente pois algumas propriedades podem exigir que estejam presentes todos os elementos do modelo para análise.

A abordagem utilizada foi a de decomposição para o contexto e que foi usada para melhorar a eficiência das técnicas de análise de modelo, envolvendo a verificação de uma sequência de invocações de operação para descobrir violações de invariantes especificados e que pode reduzir significativamente o tempo necessário para execução de uma análise. Com esta análise detalhada se tem o objetivo de melhorar a análise do contexto para aplicações cientes de contexto apoiando o desenvolvimento dos mesmos. Na Figura 5 pode-se ver um exemplo de modelo de classe que decomposto separando os elementos para análise de suas variáveis e regras.

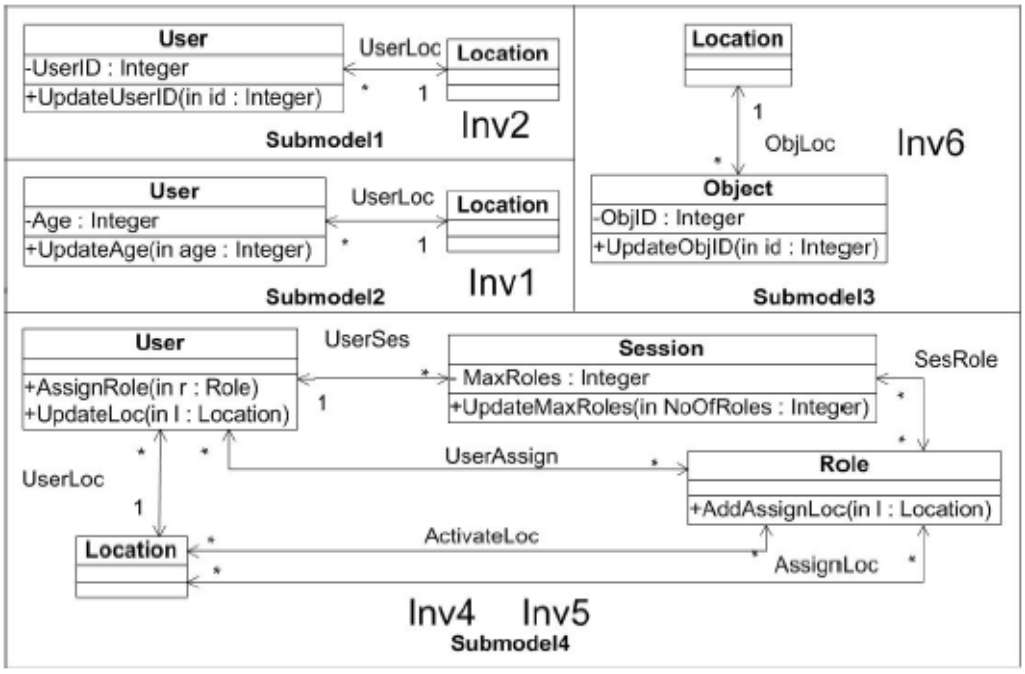
Figura 5: Modelo inicial LRBAC mostrando uma classe parcial.



Fonte: Sun, France e Ray (2013).

Na Figura 6 é mostrada uma lista de fragmentos gerados pelos algoritmos a partir do modelo LRBAC anterior. Os fragmentos foram gerados conforme os passos estabelecidos pela linguagem e os algoritmos especificados.

Figura 6: Lista de fragmentos do modelo.



Fonte: Sun, France e Ray (2013).

3.1.2 UML CAProf

Em Benselim e Seridi-Bouchelaghem (2013) foi estendida a notação de diagramas UML para o desenvolvimento para aplicações baseadas em contexto, apresentando um conjunto de novas notações específicas do diagrama de classes UML para aplicações de modelagem de ciência de contexto usando a plataforma de modelagem de software StarUML.

Como a notação padrão UML não suporta todos os aspectos do contexto de uso de forma adequada, neste trabalho foi feita uma extensão do diagrama de classes UML para representar e modelar contexto, definida por alguns mecanismos de extensibilidade que permitem a obtenção de uma representação gráfica específica de uma situação contextual. O objetivo visa facilitar a extração e modelagem de todos os elementos que podem influenciar a situação atual de um usuário inserido em um contexto. Diagramas de classes foram utilizados para descrever os diferentes tipos de contexto e seus relacionamentos.

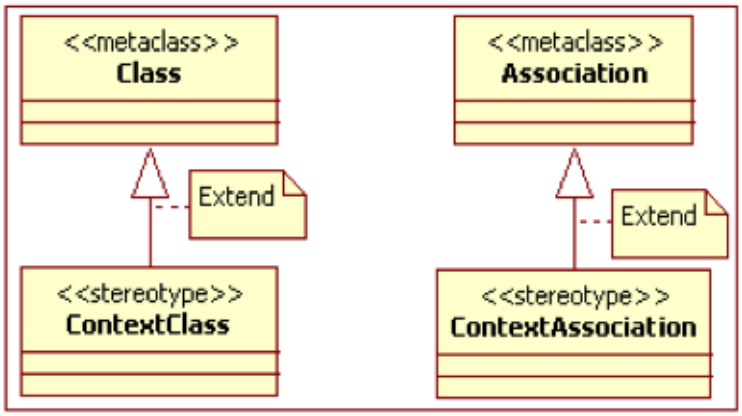
Para efetuar a modelagem através da extensão UML foi utilizado perfis UML para o domínio de consciência de contexto utilizando o conjunto de mecanismos de extensão para representar perfis de trabalho, o modelo chamado de "UML CAProf" (UML Context-Aware Profile) foi feita a partir de uma nova visão da abordagem MDA (Model Driven Architecture) que leva em conta as alterações no contexto de uso. Assim foram definidos estereótipos, valores etiquetados e restrições em uma extensão da UML padrão criando perfis. Primeiro definiu-se três restrições que foram representadas por uma notação da UML:

- Restrições relacionadas com o próprio usuário como: identidade, perfil, comportamento, preferências, etc.;
- Restrições relacionadas com a aplicação como: software, hardware, redes, modo de interação, etc.;
- Restrições relacionadas ao meio ambiente como: tempo, localização, clima, pessoas e objetos próximos, recursos disponíveis, etc.

Definiu-se assim restrições de semântica para cada elemento novo adicionado. As restrições estão ligadas aos estereótipos definidos e podem ser representadas por linguagem natural ou pela OCL (Object Constraint Language). Após, os valores etiquetados são ligados a um estereótipo com a finalidade de indicar atributos do estereótipo e são considerados como meta-atributos de metaclasses.

Estereótipos específicos são criados para comportar todos os componentes de um contexto e para garantir uma melhor representação do contexto no desenvolvimento de aplicações distribuídas. São definidos elementos contextuais com o tipo definido por uma das entidades: usuário, aplicativo, ambiente ou comportamento; Cada elemento contextual tem um estereótipo que são obtidos pela especialização do estereótipo principal. Na Figura 7 pode-se observar um diagrama exibindo as metaclasses da UML: classe e associação. Elas são estendidas para criar novos elementos UML.

Figura 7: Metaclasses da UML "Classe" e "Associação".



Fonte: BENSELIM e SERIDI-BOUCHELAGHEM (2013).

A implementação do "UML CAProf " segue os seguintes passos:

a) Preparação de arquivo de perfil UML: um arquivo XML é preparado contendo todos os componentes do perfil;

b) A criação de estereótipos: um arquivo XML é preparado contendo uma lista de estereótipos como nome, uma breve descrição e a classe UML base deste estereótipo;

c) Criação de restrições: através do "Editor de Restrição" do menu do StarUML todas as restrições propostas deverão ser introduzidas;

d) Criação de valores etiquetados: através do "Tagged Valor Editor" do menu do StarUML pode ser personalizado as propriedades necessárias através de um arquivo XML;

e) Estender (ou personalizar) sistema de menu do StarUML: o menu pode ser estendido adicionando novos itens de menus relacionados com os mecanismos de extensibilidade propostos. Adicionando um arquivo XML pelo Gerenciador de Perfis de StarUML.

Assim o perfil para desenvolver aplicações sensíveis ao contexto já está pronto para ser usado no desenvolvimento. Assim o sistema modelado nessa proposta é influenciado pelos seguintes elementos contextuais: usuário, localização, dispositivos, temperatura, rede, as pessoas próximas e os objetos ao redor.

Este modelo prevê a criação de perfis cada vez que uma modelagem foi feita onde primeiramente um arquivo XML deve ser definido com os componentes que serão utilizados na modelagem, adicionando um trabalho para os desenvolvedores de efetuar uma análise anterior especificando o que entrará no perfil que foi utilizado.

3.1.3 CAMEL

Neste trabalho Sindico e Grassi (2009) abordam o desenvolvimento dirigido para modelagem de sistemas de software sensíveis ao contexto, estudando um desenvolvimento modelo orientado para a consciência do contexto para permitir que um desenvolvedor possa lidar com preocupações de conscientização contexto na fase de concepção de um software. Para tanto desenvolvida uma linguagem de modelagem de domínio específico chamado CAMEL (Context Awareness Modeling Language) que permite enriquecer modelos UML de

forma independente para sistemas sensíveis ao contexto. Assim podem ser feitas transformações no software e estas podem ser aplicadas aos modelos CAMEL juntamente com o modelo UML definidos, com finalidade de gerar código executável para ser usado em uma plataforma específica.

CAMEL propôs uma extensão de UML que instancia o modelo de domínio conceitual para introduzir a consciência de contexto, utilizando Eclipse Modeling Framework (EMF) desenvolvendo um editor para a nova linguagem a partir de uma especificação de um metamodelo formalmente descrita no ECORE (uma linguagem proprietária integrada com o editor Eclipse UML). Nesse formato possível a introdução de modelos de recursos de sensibilização contexto em modelos TS - Target System, definidos por meio da UML sem ter que modificá-los. Nessa linguagem a consciência contexto é tratada por meio de três partes separadas:

- Sensoriamento de contexto: conjunto de atividades que fazem a leitura de informações contextuais a partir de sensores físicos ou lógicos;
- Acionando adaptação de contexto: conjunto de atividades que avaliam continuamente informações contextuais e em certas condições acionam de mecanismos de adaptação;
- Adaptação de contexto: conjunto de mecanismos de adaptação que podem ser acionados pelo item acima.

São fornecidos dois construtores para modelar informações de contexto:

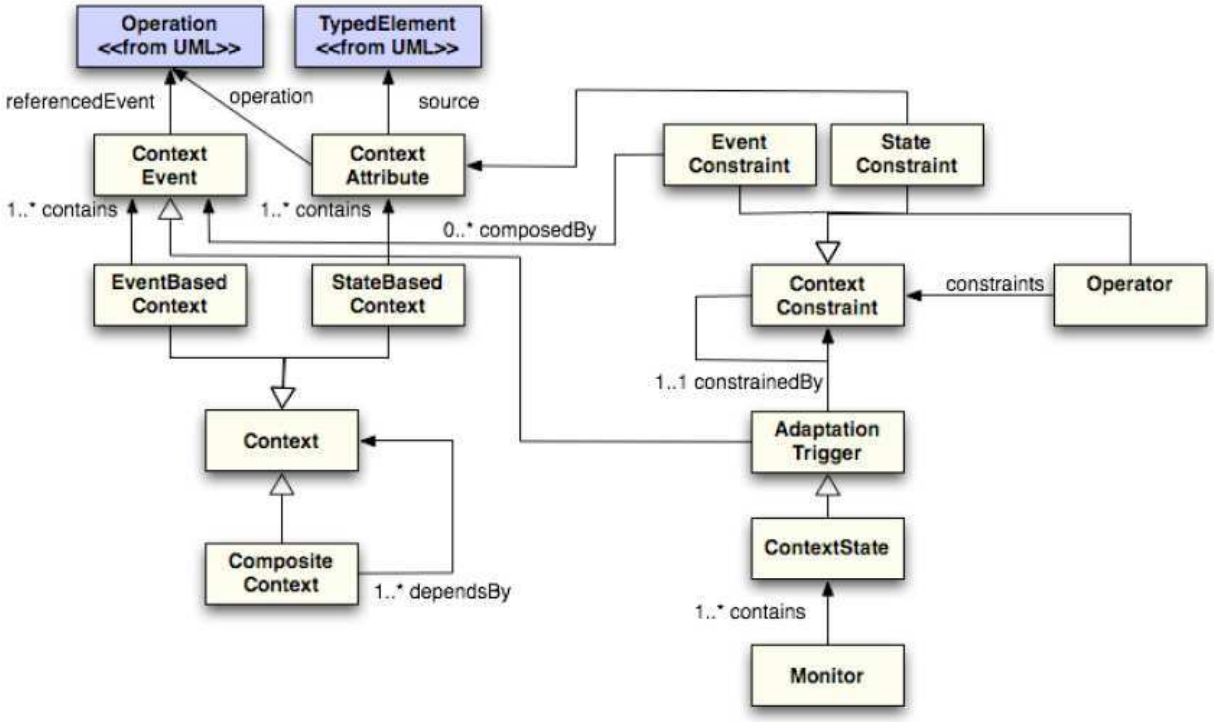
- StateBasedContext: formado por um conjunto de atributos representados pela estrutura ContextAttribute ele recebe informações contextuais estáticas;
- EventBasedContext: formado por um conjunto de eventos representados pela estrutura ContextEvent ele recebe informações contextual dinâmica.

Nesse ponto utilizado um Monitor de Construção (Construct Monitor) que é representado pela estrutura ContextState e recebe o que chamam de gatilhos de adaptação relacionados logicamente. Existe uma estrutura ContextConstraint onde estão relacionados os estado do contexto verificando a informação contextual envolvida, considerando o sistema no estado contexto relacionado. Esta estrutura está dividida em tipos: StateConstraint (restrições de estado), EventConstraint (restrições de eventos) e Operador, que permite compor restrições de estado e de eventos.

Dois conceitos foram utilizados para adaptação do contexto. Um deles trata da adaptação consciente de contexto onde mecanismos de verificação e adaptação de contexto são acionados toda vez que um ambiente diferente é percebido durante as atividades de monitoramento. E outro é a adaptação contextual pela linguagem CAMEL pode ser realizada pelos mecanismos de ligações de contextos conscientes e inserções de contextos conscientes.

No CAMEL os adaptadores são as entidades que atuam como recipiente para adaptação das camadas que estão logicamente relacionados, recebem sinais pelos monitores e ativam ou desativam as camadas de adaptação de acordo como os estados do contexto. Na Figura 8 pode-se visualizar o metamodelo Camel criado que estende os itens Operation e TypedElement, em roxo, para a representação do contexto.

Figura 8: Metamodelo CAMEL estendendo itens de contexto.



Fonte: Sindico e Grassi (2009).

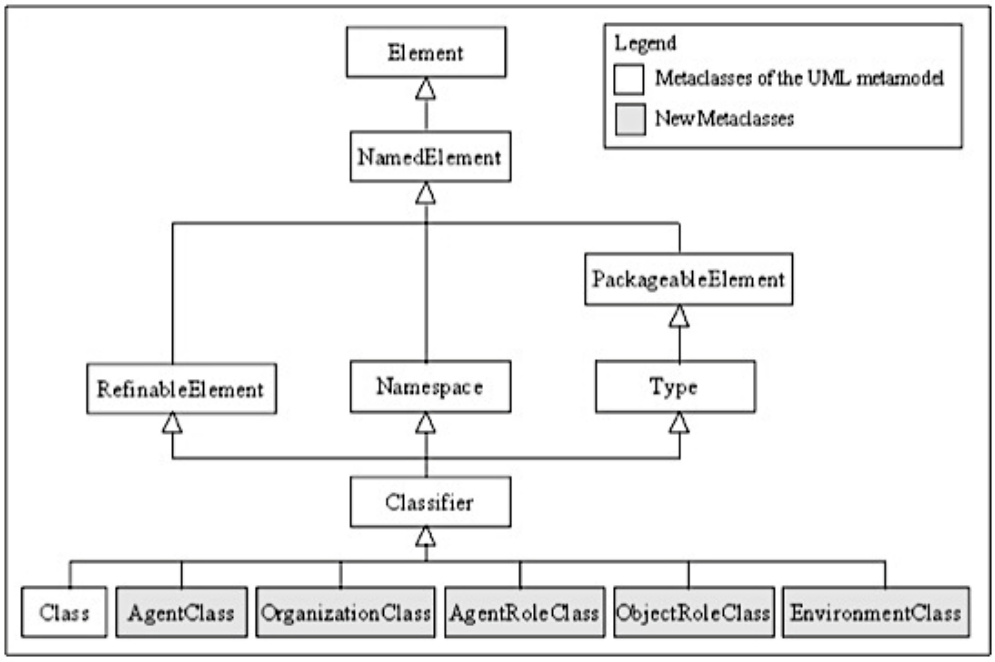
3.1.4 MAS-ML

Silva, Choren e Lucena (2008) desenvolveram uma Linguagem de Modelagem para Sistemas Multi Agentes chamada MAS-ML (Multiagent System Modelling Language) que fornece uma extensão conservadora utilizando a UML através de um metamodelo que inclui as entidades necessárias para modelar sistemas multiagentes (SMAs), por exemplo agente-relacionados, tais como papéis, organizações, planos e protocolos. Houve um esforço para montagem de um metamodelo também chamada MAS-ML que detalha novos conceitos introduzidos neste metamodelo UML para lidar com entidades orientadas para os agentes.

Novos elementos foram localizados para modelagem e trabalhados em três diagramas estruturais que são diagramas de classe estendidos, papéis e organizações e também dois diagramas dinâmicos. O artigo fornece uma linguagem de modelagem visual para especificação e modelagem para sistemas que incorporem conceitos elaborados a partir da teoria de massa para agentes, mais especificamente, a partir do metamodelo que incorpora os conceitos de TAO (Taming Agents and Objects). Integrando conceitos de agente orientados e entidades existentes no metamodelo UML e que foram estendidas para contemplar todos os objetos e comportamentos usados para o desenvolvimento de softwares que utilizam agentes inteligentes.

Foi efetuadas extensões nos diagramas UML que definiram um conjunto de conceitos orientados a agentes e efetuadas modificações para comtemplar conceitos de agentes com as metaclasses existentes ou quando necessário incluídas novas meta classes. Os conceitos trabalhados no artigo foram: agente, organização, ambiente e papel. Para cada conceito foi introduzido uma nova metaclasse que podem ser vistas na Figura 9 onde mostra as novas classes introduzidas, na cor cinza.

Figura 9: Metamodelo inicial com novas classes criadas para MAS-ML.



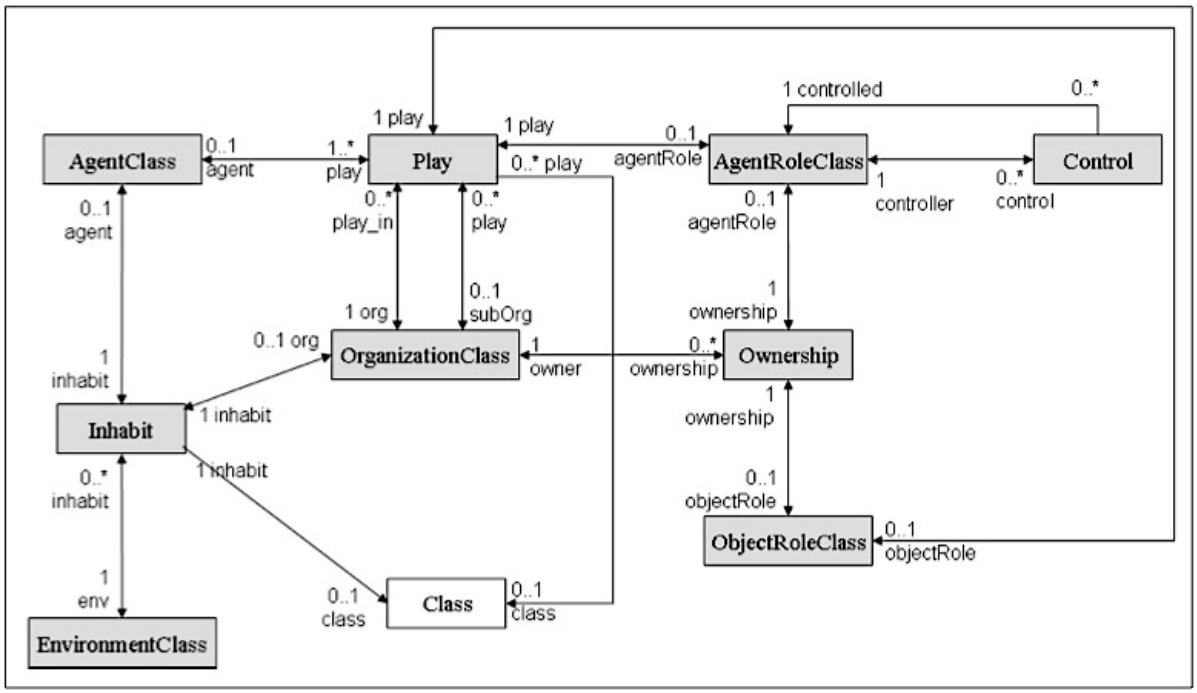
Fonte: Silva, Choren e Lucena (2008).

As metaclasses estendidas possuem as suas próprias associações e restrições diferente das fornecidas pela UML, resumidamente são elas:

- AgentClass: conjunto de agentes que compartilham as mesmas especificações das características, restrições e semântica;
- OrganizationClass: conjunto de organizações que compartilham as mesmas especificações das características, restrições e semântica. São usadas para descrever a estrutura social de um agente;
- AgentRoleClass: conjunto de funções de agentes que compartilham as mesmas especificações das características, restrições e semântica. As características estruturais são objetivos, deveres, direitos e protocolos.
- ObjectRoleClass: conjunto de recursos vistos pelas entidades que tenta acessar o objeto fazendo o papel do mesmo;
- EnviromentClass: conjunto de ambientes que compartilham as mesmas especificações das características, restrições e semântica. Fornece o contexto lógico para os agentes, organizações e objetos para executar suas ações.

A Figura 10 mostra as metaclasses desenvolvidas a partir do *"AngetClas"* para o conceito de agentes e os relacionamentos entre as mesmas. As metaclasses criadas estão destacadas em cinza. Pode-se observar através das ligações as dependências entre as metaclasses e a multiplicidade entre elas.

Figura 10: Relação entre as novas metaclasses incluídas no MAS-ML.



Fonte: Silva, Choren e Lucena (2008).

Para definir as características estruturais dos sistemas multiagentes foi incluindo dois estereótipos dos sistemas multiagentes:

- Objetivos: uma característica da estrutura de uma classe de agente que especifica a definição do seu objetivo ou papel;
- Crença: uma característica da estrutura de uma classe de agente que especifica a definição de propriedade.

Para definir as características comportamentais dos sistemas multiagentes para cada conceito foram definidas metaclasses:

a) ActionClass: descrever um conjunto de ações de um agente ou uma classe de organização;

b) PlanClass: uma característica comportamental, descreve um conjunto de ações a serem executadas para atingir os objetivos.

c) ProtocolClass: uma característica comportamental, descreve as interações entre papéis através da especificação das mensagens;

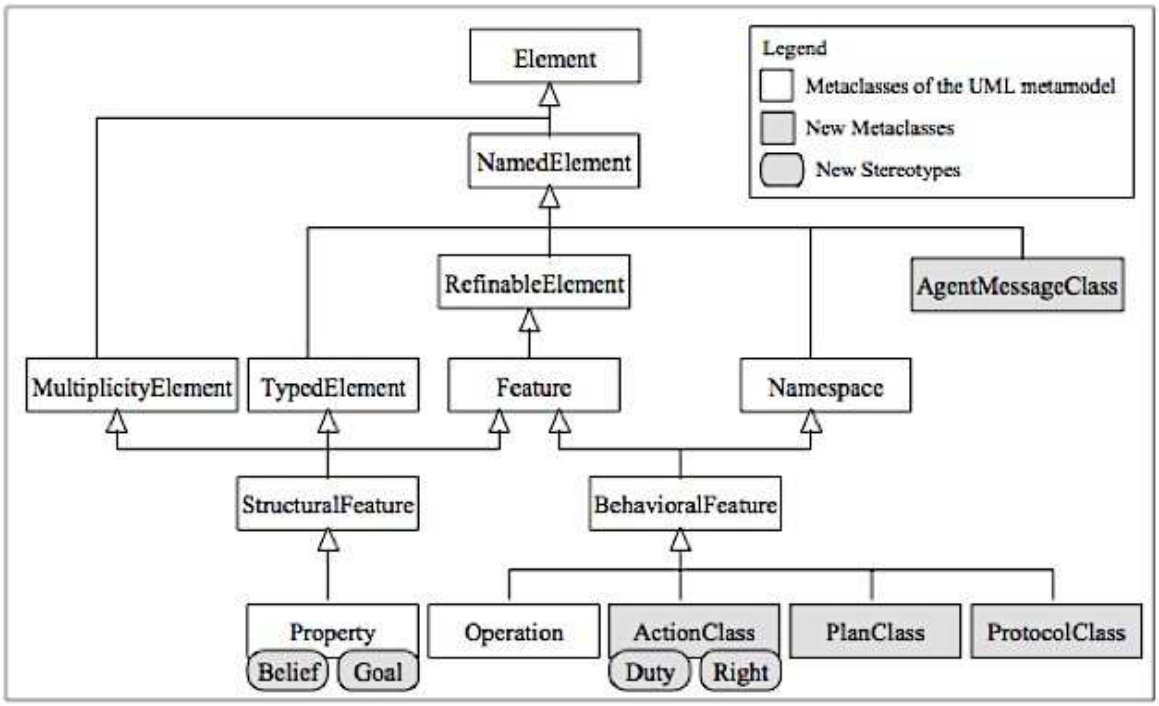
d) AgentMessageClass: um elemento nomeado, que define as mensagens trocadas pelos agentes.

E também foram definidos dois estereótipos:

- Dever: identifica as ações que devem ser executadas pelo agente que está fazendo um papel específico;
- Direito: identifica ações que o agente pode executar no papel em que está descrevendo as permissões associadas as ações.

Na Figura 11 pode ser visualizado um metamodelo mostrando as características de comportamento das novas metaclasses incluídas para o MAS-ML.

Figura 11: Metamodelo com novas características de comportamento do MAS-ML.



Fonte: Silva, Choren e Lucena (2008).

Utilizando os conceitos da UML foram definidas novas relações dos sistemas multiagentes que referencia os elementos relacionados, conforme o metamodelo da Figura 9:

a) Ownership: propriedade que especifica uma relação semântica entre instâncias de um papel e de uma organização;

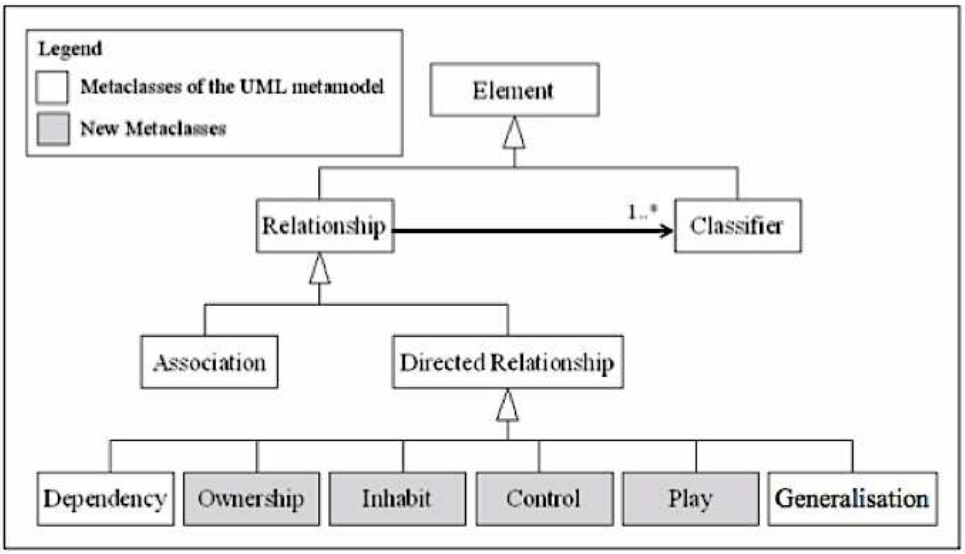
b) Inhabit: especifica uma relação semântica que pode ocorrer entre instâncias de entidades em um ambiente;

c) Control: especifica uma relação semântica específica entre os casos de funções de agentes. Um controle é usado para modelar um relacionamento social do tipo mestre-escravo entre as funções de um agente;

d) Play: relação semântica entre instâncias de um agente ou objeto, de um papel e de uma organização.

Na Figura 12 pode ser visualizada estas novas relações detalhadas.

Figura 12: Metamodelo mostrando as novas relações.



Fonte: Silva, Choren e Lucena (2008).

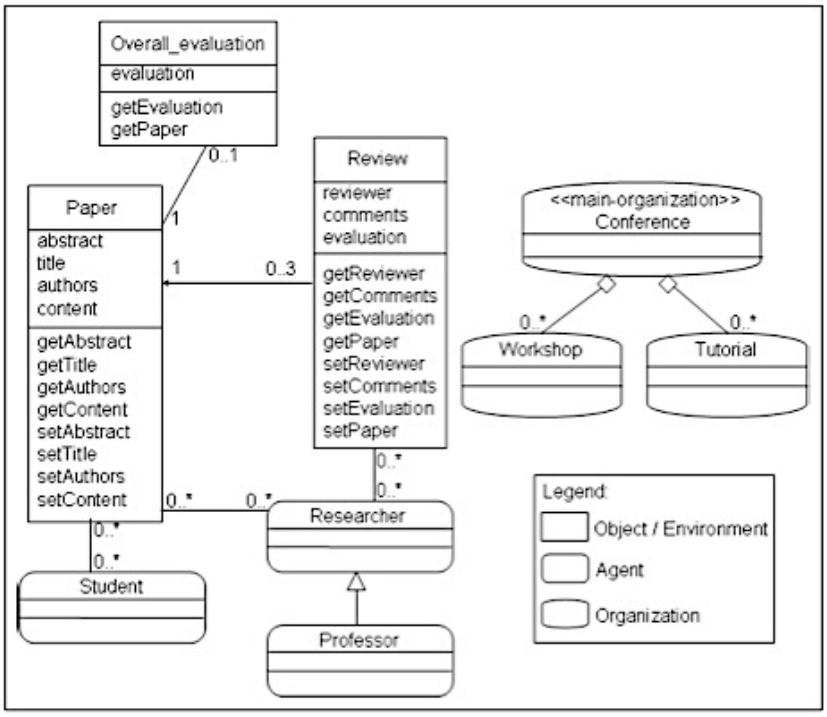
Assim foram feitos três diagramas estruturais para modelar os aspectos estáticos:

a) Diagrama de classes UML estendida: o diagrama de classes MAS-ML que é uma extensão do diagrama de classe UML que modela aspectos estruturais das entidades classes, agentes, organizações, ambientes e as relações entre elas. Na Figura 13 pode-se visualizar um exemplo de um diagrama de classe do sistema de Gestão de Conferência (DeLoach, 2002) utilizado por Silva, Choren e Lucena (2008) para modelo, mas que ilustra a utilização dos conceitos levantados na linguagem.

b) Esquema de organização: expressada nos modelos das organizações do sistema detalhando como as organizações estão relacionados com as outras entidades do sistema. Na Figura 14 pode-se visualizar um exemplo de um diagrama de organização do sistema de Gestão de Conferência (DeLoach, 2002) utilizado por Silva, Choren e Lucena (2008) para modelo.

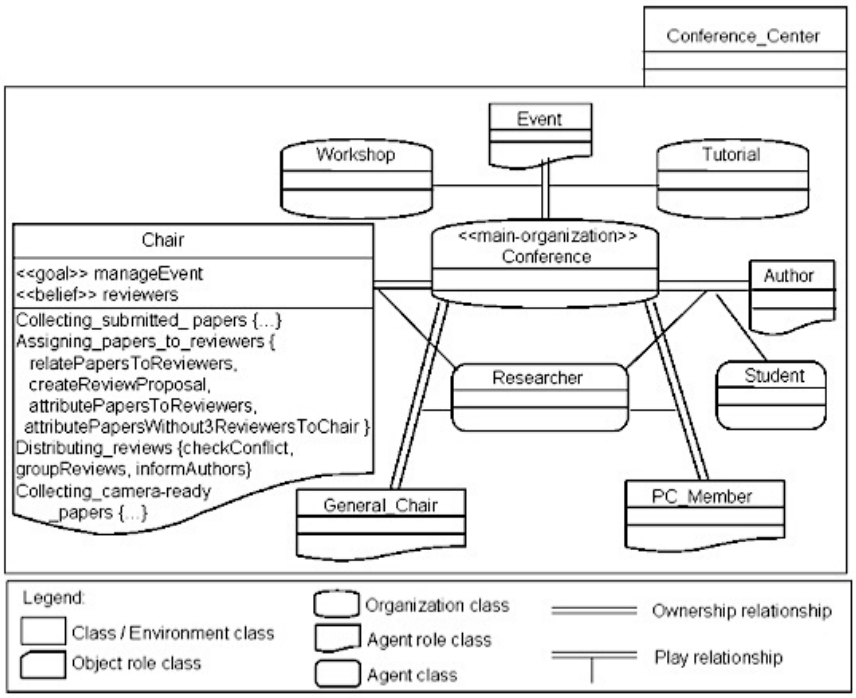
c) Diagrama de papel: é responsável por modelar os aspectos estruturais de funções de agentes e papéis definidos nas organizações, assim como as relações entre os papéis e os objetos no ambiente. O diagrama de papel do sistema de gestão de conferência (DELOACH, 2002), na Figura 15 mostra as classes de funções de agente e também as classes de funções do objeto definido no que chamam organização da conferência do sistema e que foi utilizado como modelo por Silva, Choren e Lucena (2008).

Figura 13: Exemplo de diagrama de classe de sistema de gestão de conferência.



Fonte: Silva, Choren e Lucena (2008).

Figura 14: Diagrama de organização do sistema de gestão (DeLoach, 2002).



Fonte: Silva, Choren e Lucena (2008).

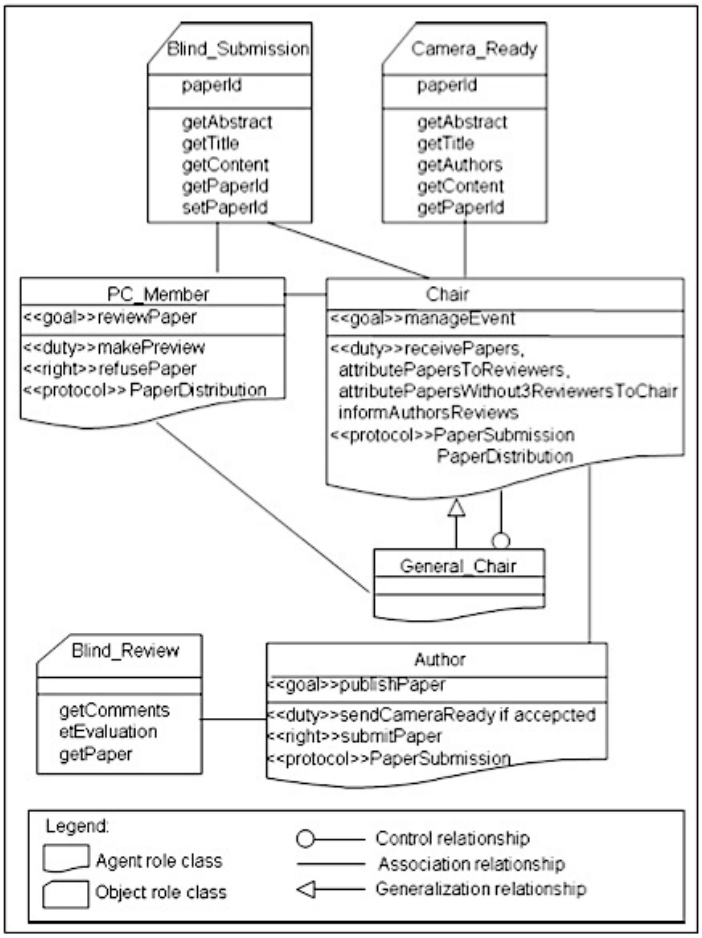
Também foram feitos dois diagramas dinâmicos diferentes:

- Diagrama de Sequência UML: estende o diagrama de sequência UML para modelar a interação entre os objetos, agentes, organizações e ambientes,

descrevendo a interação entre objetos e a identificação dos papéis desempenhados por eles.

- Diagramas de atividade: estende o diagrama de atividade UML para modelar os planos e ações dos agentes, organizações e conceitos.

Figura 15: Diagrama de papel do sistema de gestão de conferência



Fonte: Silva, Choren e Lucena (2008).

3.1.5 Comparação dos Trabalhos Relacionados

Esta seção apresenta uma análise comparativa entre os trabalhos relacionados e o modelo proposto. É apresentado então um resumo dos principais itens entre os trabalhos analisados na Tabela 3. Na coluna “Comparativo” foram incluídos critérios de comparação importantes e que serão abordados no modelo proposto.

Para a comparação foram elencados critérios relevantes no contexto e que foram motivação desta pesquisa:

- *Estende a UML:* avalia se a UML foi estendida no artigo para contemplar itens não convencionais;
- *Linguagem:* identifica a linguagem de definição do metamodelo;

- *Modelagem:* identifica o framework de modelagem utilizado;
- *Ferramenta:* identifica se foi utilizada alguma ferramenta para representação dos modelos dos artigos;
- *Implementação:* identifica o que foi feito de implementação para os modelos que foram propostos nos artigos;
- *Diagramas trabalhados:* identifica os diagramas UML que foram utilizados;
- *Abordagem:* identifica as abordagens utilizadas para geração dos modelos propostos nos artigos.

Assim a Tabela 3 mostra um consolidado com os critérios levantados para cada um dos trabalhos relacionados citados neste capítulo.

Tabela 3: Tabela de comparação entre os trabalhos relacionados.

Comparativo	**Método de Divisão**	**UML CAProf**	**CAMEL**	**MAS-ML**
Estende UML	Não estende	Perfis	Operation TypedElement	AgentClas e outros relac.
Linguagem	OCL	OCL	ECORE	OCL
Modelagem	EMF	MDA	EMF	SMA
Ferramenta	Kermeta	StarUML	Eclipse	Não utilizou
Implementação	Protótipo de análise	Inclusão de perfil para modelagem	Linguagem de modelagem	Linguagem de modelagem
Diagramas trabalhados	Classe	Classe	Classe	Classe e Sequência
Abordagem	Particiona o modelo	MDA	UML estendido	TAO

Fonte: Elaborado pela autora.

No Método de Divisão de Sun, France e Ray (2013) foi proposto uma modelagem fatiando o contexto com algoritmos, mas seu objetivo era chegar em uma técnica de análise de modelo, envolvendo a verificação de uma sequência de invocações de operações para descobrir violações de invariantes especificadas reduzindo o tempo de análise. Essa técnica de análise de contexto somente engloba a forma com que o contexto deve ser abordado e quais as

variáveis que devem ser consideradas, não chega a efetuar uma extensão UML para comportar graficamente sua modelagem para auxílio direto no desenvolvimento.

Na UML CAProf de Benselim e Seridi-Bouchelaghem (2013) foi estendido a modelagem de contexto através de perfis inseridos por XML no StarUML, onde cada perfil se limita um estereótipo para uma configuração de contexto. Nessa proposta limitou-se o perfil ao seu estereótipo, estendendo apenas classe e associação.

No CAMEL por Sindico e Grassi (2009) é feita uma extensão de UML para sistemas sensíveis ao contexto em Eclipse utilizando o ECORE, porém a modelagem de sistemas sensíveis ao contexto é limitada por camadas de trabalho separadas em sensoriamento, acionamento e adaptação trabalhando o relacionamento entre elas e não entrando em detalhes sobre o contexto e seus componentes de análise. Foram estendidos os itens Operation e TypedElement.

Na linguagem MAS-ML Silva, Choren e Lucena (2008) desenvolveu uma linguagem de modelagem com detalhes sobre os componentes envolvidos e suas relações, estendendo a UML para comportar esses novos objetos e suas relações como agente, organização, papeis, objeto e comportamento. No entanto esta linguagem foi desenvolvida voltada para sistemas que vão trabalhar com a abordagem de agentes inteligentes.

No UML2Contex, propõe-se uma extensão da UML para modelagem de contexto para auxiliar no desenvolvimento de sistemas cientes de contexto. Aprofunda-se nos conceitos e nos termos utilizados relevantes para contexto. Procurou-se nas ontologias o melhor vocabulário para utilização não limitando o desenvolvedor na criação de perfis para modelar o contexto e já prevendo a inclusão de variáveis que serão consideradas na modelagem contexto. O modelo proposto difere dos trabalhos relacionados nos seguintes pontos: não foi limitado por camadas de trabalho, não aplicou algoritmos para quebra de modelagem, abordará de maneira visual a modelagem de contexto facilitando o trabalho dos desenvolvedores e definirá uma linguagem para padronização dos termos utilizados. Um trabalho foi feito por Silva, Choren e Lucena (2008) onde foi desenvolvido uma linguagem de modelagem, no entanto focou em agentes inteligentes e no UML2Context foi abordado especificamente o contexto para sistemas cientes de contexto.

4 UML2CONTEXT

Após apresentar os principais conceitos exigidos para o entendimento completo do trabalho, bem como identificar as semelhanças e as diferenças do trabalho proposto em relação à literatura atual sobre modelagem de contexto, este capítulo tem como objetivo apresentar a extensão da UML proposta para modelagem de contexto, nomeada de *UML2Context*. Para isso, uma extensão da UML foi trabalhada com o objetivo de inserir no metamodelo da UML os principais conceitos para a modelagem de contexto. Seguindo esta abordagem, foi possível estender a UML e permitir modificar o metamodelo padrão sem afetar qualquer conceito previamente definido na linguagem, dado que a mesma seguirá os princípios de projeto *Open-Closed*, que preconiza que toda modificação deve ser dominada por adição e não por alteração dos elementos existentes (MARTIN, 2002).

A UML pode ser estendida de duas maneiras, com um novo dialeto UML ou nova linguagem UML. O *UML2Context* utiliza a segunda opção reutilizando parte do pacote InfrastructureLibrary e aumentando-o com estereótipos como metaclasses e metarelationships apropriadas especificando uma nova linguagem. Assim foram criados então a *UML2Context Language* e a *UML2Context Tool*, utilizando os mecanismos de extensibilidade da UML: estereótipos, valores atribuídos e restrições. Foram utilizados diagramas de classe para representação. Os diagramas de classe são geralmente utilizados para demonstrar a extensão efetuada da UML, pois mostram atributos e operações de uma classe e as restrições e a maneira como os objetos são conectados (FOWLER; SCOTTS, 2000).

É importante ressaltar as perspectivas com que foram abordados os diagramas de classe. Existem 3 perspectivas que podem ser usadas ao criar um digrama de classe: conceitual, especificação e implementação (FOWLER; SCOTTS, 2000). Na perspectiva conceitual projeta-se um diagrama que representa os conceitos no domínio que está sendo usado, e que serão naturalmente relacionados às classes, e que neste trabalho inicia-se o diagrama de classes com essa perspectiva introduzindo os conceitos que evidenciamos do domínio contexto; Na perspectiva especificação, examinamos o software analisando as suas interfaces e não sua implementação, não faremos esta análise neste trabalho; Na perspectiva implementação, realmente temos classes e estamos expondo a implementação do sistema modelado, esta perspectiva será abordada na ferramenta onde define-se e visualiza as classes.

Este capítulo é organizado da seguinte forma. A seção 4.1 apresenta os conceitos adicionados à UML para modelagem de contexto e como os elementos foram estereotipados para extensão da UML. A seção 4.2 descreve o padrão de descrição de metamodelagem utilizado para descrever a extensão proposta. A seção 4.3 apresenta a extensão proposta. Por fim, a seção 4.4 discute como a extensão proposta foi instanciada.

4.1 Definição dos Conceitos da Extensão

Definindo a notação que é a parte gráfica, se estabelece a sintaxe da linguagem de modelagem, que define como os elementos e o conceito de contexto foram modelados. Utilizando os mecanismos de extensibilidade são definidos os estereótipos, que estendem o vocabulário UML, permitindo a criação de novos tipos de blocos derivados dos já existentes, mas que serão específicos ao para o contexto (FOWLER; SCOTTS, 2000).

Baseado na análise dos trabalhos relacionados sobre ontologias de contexto, o conceito de contexto pode ser representado considerando os seguintes elementos em sua composição: Usuário, Localização, Dispositivo, Atividade e Tempo. Entende-se que estes

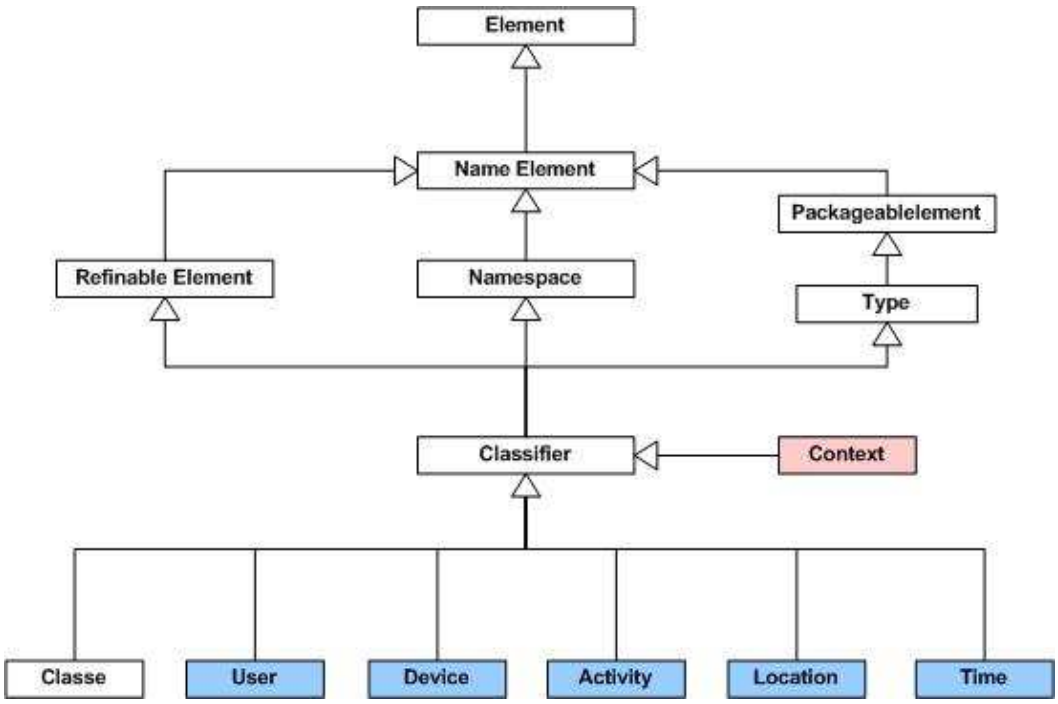
conceitos não esgotam a forma de representar Contexto, pelo contrário, eles representam um conjunto de termos que são essenciais para uma boa comunicação e documentação do contexto durante o desenvolvimento de um sistema ciente de contexto. É importante destacar que tais conceitos foram escolhidos a partir dos seguintes trabalhos, Bulcão (2006), Bulcão e Pimentel (2005), Gu et al. (2004), Strang e Linnhoff-Popien Dey (2004), Schmidt, Beigl e Gellersen (1999).

4.1.1 Metaclass

Metaclasse é um estereótipo que estende o vocabulário da UML, permitindo a criação de novos tipos de blocos de construção derivados dos já existentes, mas específicos ao seu problema (BOOCH et al., 2000). Cada um dos elementos definidos é um tipo especial de classe e pode ser chamada de estereótipo de classe ou um elemento estereotipado. Um estereótipo pré-definido na UML que foi utilizado é a *metaclass*, esse estereótipo é aplicado a uma classe cujas outras classes são instâncias e devem estar de acordo com a metaclasse. A metaclasse especifica um classificador cujos objetos são todas classes (BOOCH et al., 2000).

Os conceitos definidos então foram utilizados em inglês e seguem: *Context*, *Location*, *User*, *Device*, *Activity*, *Time*. Entendemos que estes são conceitos-chave para o contexto no desenvolvimento de software ciente de contexto, merecendo ser conceitos de primeira classe nos modelos UML2Context. A Figura 16 ilustra as metaclasses criadas para representar os conceitos definido nas ontologias, para cada conceito um elemento foi estereotipado como uma metaclasse, que estão destacadas na cor azul e foram inseridos no metamodelo da UML criando o próprio metamodelo. Na figura pode-se ver que as metaclasses definidas possuem o objetivo de especificar o contexto que é uma metaclasse mais ampla, um classificador. Abaixo dele estão os elementos que o compõem.

Figura 16: Metamodelo UML2CONTEXT com novas metaclasses inseridas.



Fonte: Elaborado pela autora

4.1.2 Responsabilidades

Cada elemento definido como uma metaclasse possui *responsabilidades* dentro do metamodelo. *Responsabilidades* são um contrato ou obrigações de uma determinada classe (BOOCH et al., 2000). Ao fazer a modelagem das metaclasses temos que especificar as responsabilidades dos elementos existentes no vocabulário estabelecido. As responsabilidades são exemplos de estereótipos quando as utilizamos em *notas*, pois a UML especifica um estereótipo-padrão aplicado às notas, os *requisitos*; Esse estereótipo dá nomes a uma categoria básica de notas utilizadas para estabelecer alguma responsabilidade ou obrigação ao elemento (BOOCH et al., 2000). Sendo assim seguem as responsabilidades definidas para cada um dos elementos:

- ***Context***: é responsável por englobar e tratar as informações das metaclasses;
- ***User***: é responsável pelo conhecimento acerca das informações sobre o usuário final que está acessando o sistema e suas características relevantes;
- ***Device***: é responsável pelo conhecimento acerca do equipamento que está sendo usado como telefone, computador e *gadgets*;
- ***Activity***: é responsável pelo conhecimento acerca da ação ou evento que está sendo efetuado, como uma compra, uma consulta, uma dica;
- ***Location***: é responsável pelo conhecimento acerca do local físico onde está ocorrendo as atividades ou ações do usuário e da localização do software, localização geográfica, ambiente, setor;
- ***Time***: é responsável pelas informações de tempo, data e hora que ocorreu o evento, duração e outros.

4.1.3 Atributos

Estes elementos escolhidos para compor o contexto também possuem informações importantes que serão consideradas pelos sistemas cientes de contexto. Essas informações são os *atributos* da classe que também contém operações próprias. Um *atributo* é uma propriedade nomeada de uma classe que descreve um intervalo de valores que as instâncias da propriedade podem apresentar (BOOCH et al., 2000). Sendo assim, segue a lista de atributos dos elementos que definimos:

- ***User*** - Atributos: nome, idade, sexo, estado civil, quantidade de filhos, endereço, cidade, estado, país, profissão, religião;
- ***Device*** - Atributos: aparelho, modelo, sistema operacional, conexão;
- ***Activity*** - Atributos: ação, informação fornecida, duração;
- ***Location*** - Atributos: local, tipo, posição geográfica, temperatura do ambiente;
- ***Time*** - Atributos: data, hora inicial, hora final, duração.

4.1.4 Operações

As operações são processos que a classe sabe realizar, que correspondem a métodos em uma classe, sendo que, uma operação é algo que é executado em um objeto enquanto o método é o corpo do procedimento (FOWLER; SCOTTS, 2000).

As metaclasses receberem as informações coletadas do ambiente e vão informar a metaclasse contexto essas informações. Sendo assim teremos para cada metaclasse os métodos:

- Leitura (read): que coleta e lê um valor de um atributo;
- Modificação (modify): que põe o valor no campo, alimentando os atributos;
- Informa (inform): que informa o valor do atributo quando consultado.

4.1.5 Relacionamentos

Os relacionamentos definem como os elementos identificados pelo vocabulário vão relacionam-se entre si. Na modelagem orientada a objetos temos três tipos de relacionamentos que fornecem uma forma diferente de combinações de abstrações: *dependências*, que representam relacionamentos de utilização entre as classes; *generalizações*, que relacionam classes generalizadas as suas especializações; e *associações*, que representam relacionamentos estruturais entre objetos (BOOCH et al., 2000).

Associações representam as relações entre ocorrências de classes, e da perspectiva conceitual representam relações conceituais entre classes. As construções básicas de associação, de atributo e de generalização fazem muito para especificar restrições importantes (FOWLER; SCOTTS, 2000).

Temos dois tipos de relacionamento estruturais (LARMAN, 2000):

- Agregações: define que um objeto possui outros objetos, e todos existem mesmo não estando associados;
- Composições: são agregações mais fortes, onde os objetos que compões um objeto maior não existem sozinhos. Se o objeto é composto de outros e ele for excluído os seus componentes são excluídos também.

Como o Context é composto pelos elementos User, Device, Activity, Location e Time, eles só existem se o Contexto existir formando uma associação de Composição. As associações já impõem restrições quando definimos nas mesmas, limites inferiores e superiores, quando uma ponta de associação tem multiplicidade ou quando direcionamos a navegabilidade como unidirecionais ou bidirecionais.

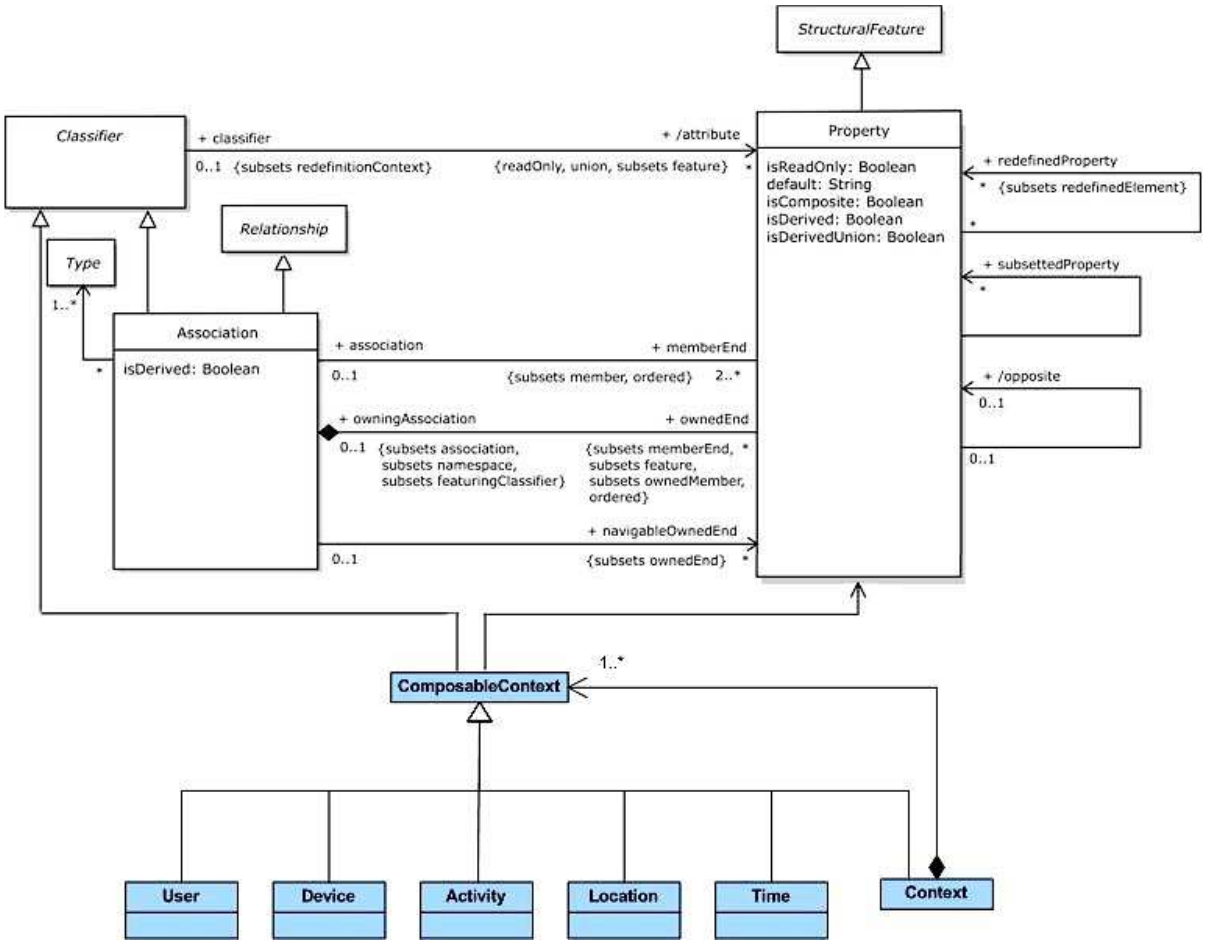
Sendo assim segue as associações entre as metaclasses e suas restrições para cada associação. Para cada elemento do contexto conterá um relacionamento da metaclasse contexto com as outras. Os relacionamentos que existirão são os seguintes:

- *Relação Context-User*: associação da metaclasse Context com a metaclasse User.
  - Restrições: cada Contexto terá um ou mais Usuários.
- *Relação Context-Device*: associação da metaclasse Context com a metaclasse Device;
  - Restrições: cada Contexto terá um ou mais Dispositivos.
- *Relação Context-Activity*: associação da metaclasse Context com a metaclasse Activity.
  - Restrições: cada Contexto terá uma ou mais Atividades.

- *Relação Context-Location*: associação da metaclasse Context com a metaclasse Location.
  - Restrições: cada Contexto terá um Localização.
- *Relação Context-Time*: associação da metaclasse Context com a metaclasse Time.
  - Restrições: cada Contexto terá um Tempo.

Os relacionamentos do contexto com os elementos são representados no metamodelo UML conforme demostrado na Figura 17. A parte superior da figura apresenta o modelo padrão de objetos UML e a parte inferior mostra em azul as metaclasses inseridas na estrutura UML.

Figura 17: Composição de relação do metamodelo.



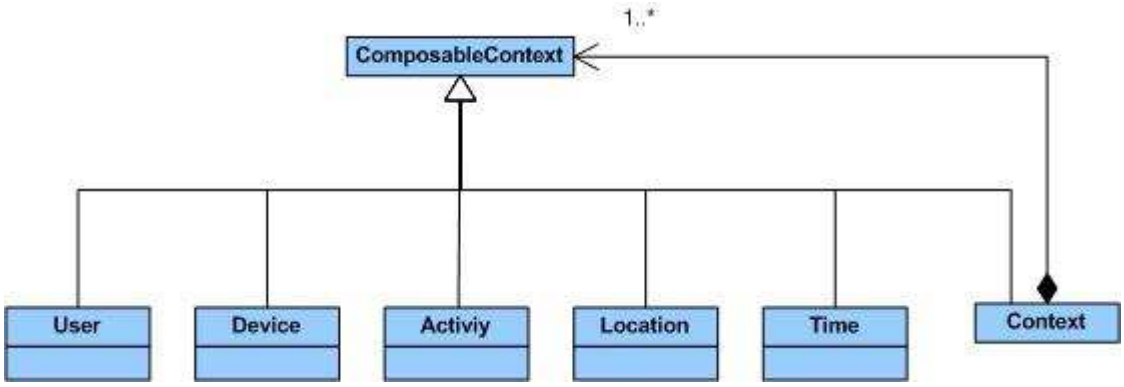
Fonte: Elaborado pela autora a partir do modelo de classe e relacionamento padrão da UML.

4.2 Extensão da UML Proposta

Com a notação definida estabelecemos o metamodelo, diagrama que define a notação, e que geralmente é utilizado um diagrama de classe (FOWLER; SCOTTS, 2000). Na Figura 18, que amplia as metaclasses da Figura 17, é possível visualizar o metamodelo proposto com

as metaclasses inseridas. Aqui vemos com melhor clareza que toda a informação que descreve o contexto de usuário é representada pelo estereótipo *ComposableContext*. O elemento *ComposableContext* é uma classe abstrata e subclasse do elemento nomeado (*Namespace* da UML), que é um elemento central no metamodelo UML. Estes elementos determinam a semântica dos elementos do modelo, que podem ser utilizados para representar contexto.

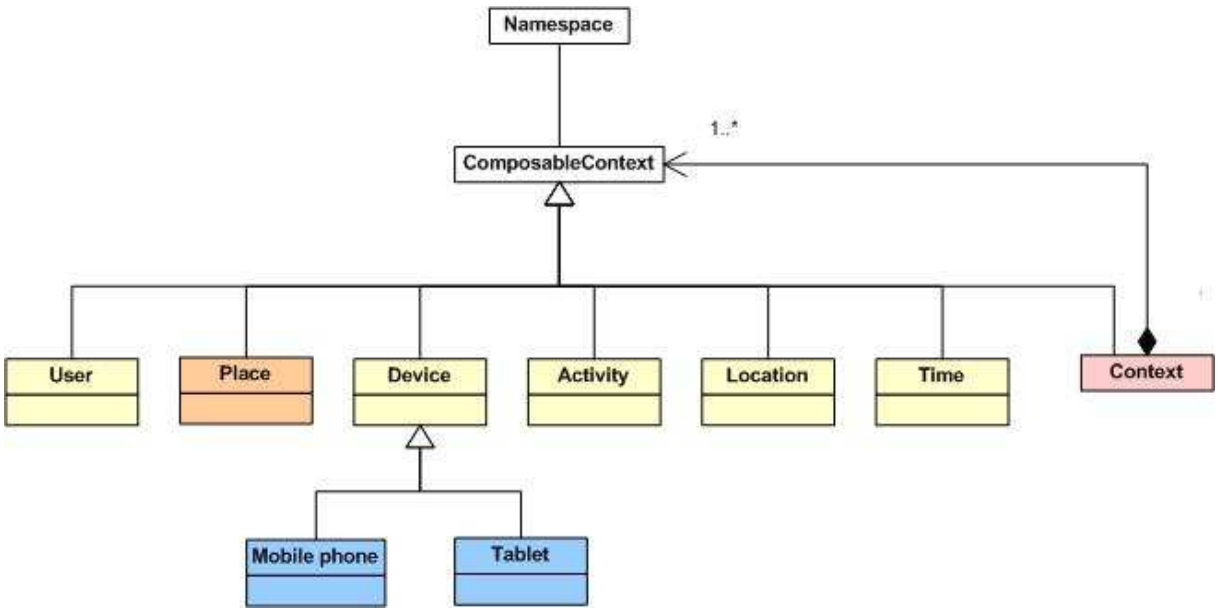
Figura 18: Novos elementos do metamodelo.



Fonte: Elaborado pela autora.

Estes elementos fazem parte da definição de contexto *ComposableContext*, que é um tipo *Classificador*. A direita do diagrama é possível ver o elemento *Context* no metamodelo, ele é composto pelas entidades que estão no mesmo nível que ele e pode conter também ele mesmo, característica que pode futuramente ser trabalhada para incluir o controle de trilhas de contexto.

Figura 19: Metamodelo com a inclusão de novos elementos.



Fonte: Elaborado pela autora.

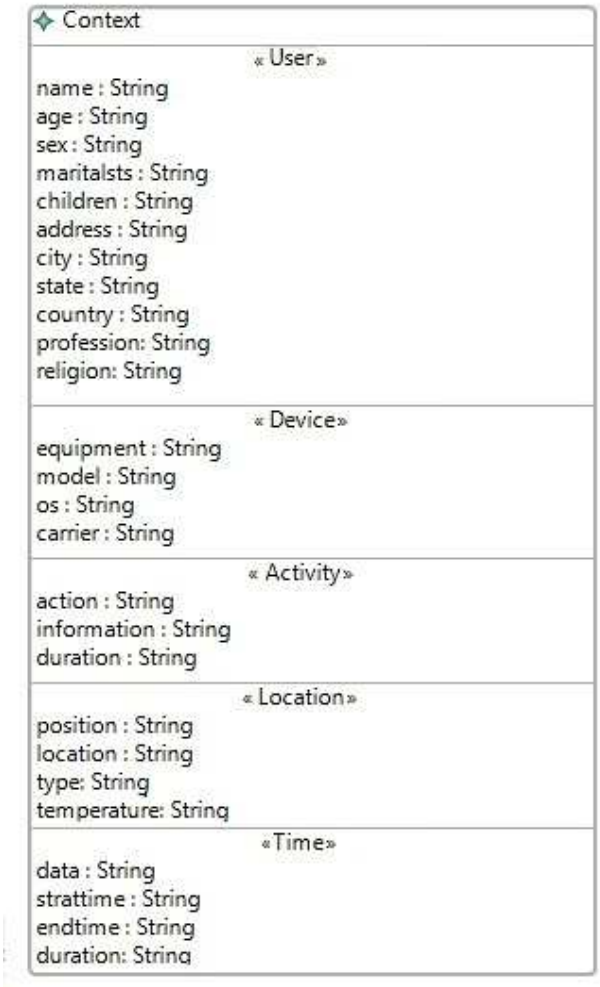
Este modelo também serve para que seja possível ampliá-lo uma vez que dependendo da aplicação poder haver necessidade de incluir outros elementos relevantes ao contexto. Assim o modelo pode ser estendido incluindo novos elementos no primeiro nível. Pode-se ver

na Figura 19 o exemplo da inclusão do elemento “Place”, onde pode ser adicionado novas informações sobre o Lugar conforme necessidade. O metamodelo proposto também pode incluir a *generalização*, que são relacionamentos entre elementos gerais chamados superclasses ou classes de primeiro nível e tipos mais específicos desses elementos chamados subclasses ou classes-filhas. Pode-se por exemplo incluir o *Device* que pode ser especializado em outros elementos que serão derivados da mesma classe como *Mobile Phone* e *Tablet*.

4.3 Exemplo de Uso da Extensão Proposta

O metamodelo define a semântica de como os elementos do modelo se instanciam em um modelo e também pode ser exemplificado por uma metaclasse. Um exemplo de como pode-se fazer uso desse metamodelo em aplicações cientes de contexto é mostrado na Figura 20, apresentando a metaclasse contexto incluindo outros elementos. Foi estendida a UML para expressar a relação de composição em tempo de modelagem. Esta relação de composição pode ser então resumida na elemento *Context* que é composto de metaclasses com suas características próprias. Esse exemplo mostra os principais elementos: user, device, activity, location e time. Cada um possui suas características como “*name*”, “*age*”, “*type*”, etc.

Figura 20: Metaclasse Context no UML2Context Tool



Fonte: Elaborado pela autora.

5 IMPLEMENTAÇÃO

Esta seção tem como objetivo descrever como a extensão proposta foi implementada, apresentar os principais *plug-in*s da plataforma Eclipse que foram utilizados e descrever o protótipo desenvolvido.

Na seção 5.1 destacaremos as tecnologias utilizadas e como se integram. Na seção 5.2 mostraremos o processo com todos os passos que foram seguidos para o desenvolvimento da ferramenta. Na seção

5.3 falaremos sobre itens que compõe a ferramenta para passar uma visão geral da mesma. E por fim na seção 5.4 expomos as vantagens e desvantagens da ferramenta até o momento.

5.1 Tecnologias Utilizadas

O UML2Context *tool* trata-se de um ambiente de modelagem específico de domínio que atenderá a modelagem de contexto de acordo com a especificação do metamodelo definido. O objetivo desta ferramenta é fornecer um ambiente de modelagem em que desenvolvedores possam trabalhar com os conceitos do domínio do problema ao mesmo tempo que utilizam explicitamente conceitos definidos no metamodelo de contexto, neste caso os conceitos e abstrações definidos no modelo de contexto proposto. Assim, é possível reduzir a problemática de representar o contexto. É importante destacar que o ambiente foi implementado como um *plug-in* da plataforma Eclipse (ECLIPSE, 2015). Os elementos que serão definidos na ferramenta *UML2Context* se baseiam no metamodelo proposto nos capítulos anteriores.

O desenvolvimento da ferramenta seguirá o processo definido pelo *plug-in* do Eclipse GMF. Além disso, os seguintes *plug-in*s serão também utilizados ECLIPSE (2015):

- EMF (*Eclipse Metamodel Framework*): um framework de modelagem e de geração de código para construir aplicações baseadas em modelos (ECLIPSE, 2015);
- EMOF (*Essencial Meta-Object Facility*): linguagem de definição de metamodelo usada para definições da UML que é parte integrante do *plug-in* EMF (ECLIPSE, 2015);
- GMF (*Graphical Modeling Framework*): *plug-in* do eclipse que fornece um componente generativo e uma infraestrutura runtime para desenvolver editores gráficos baseado no EMF e GEF (ECLIPSE , 2015);
- GEF (*Graphical Eclipse Framework*): fornece tecnologia para criar ricos editores gráficos no Eclipse (ECLIPSE, 2015).

Com o *plug-in* GMF é feita a integração entre o EMF e GEF, que é fundamental na criação do ambiente de modelagem. Logo isso permitirá aos usuários utilizarem os recursos oferecidos pela plataforma Eclipse ao mesmo tempo que fazem uso da UML2Context *tool*. Dado que muitas plataformas móveis e ubíquas são implementadas em Java, o uso da plataforma Eclipse também pode facilitar uma possível geração de código dentro do mesmo ambiente de desenvolvimento que pode ser feita posteriormente. O processo de desenvolvimento da ferramenta pode ser resumido pela necessidade de mitigar dois problemas: o domínio do problema (modelagem de contexto) ao domínio da solução (a ferramenta UML2Context). Onde a partir do metamodelo do contexto, deseja-se derivar a

ferramenta de modelagem, utilizando uma abordagem de desenvolvimento dirigido por modelos, onde o modelo central e de maior abstração é o próprio metamodelo do contexto que efetuará a geração de modelos gráficos de entendimento.

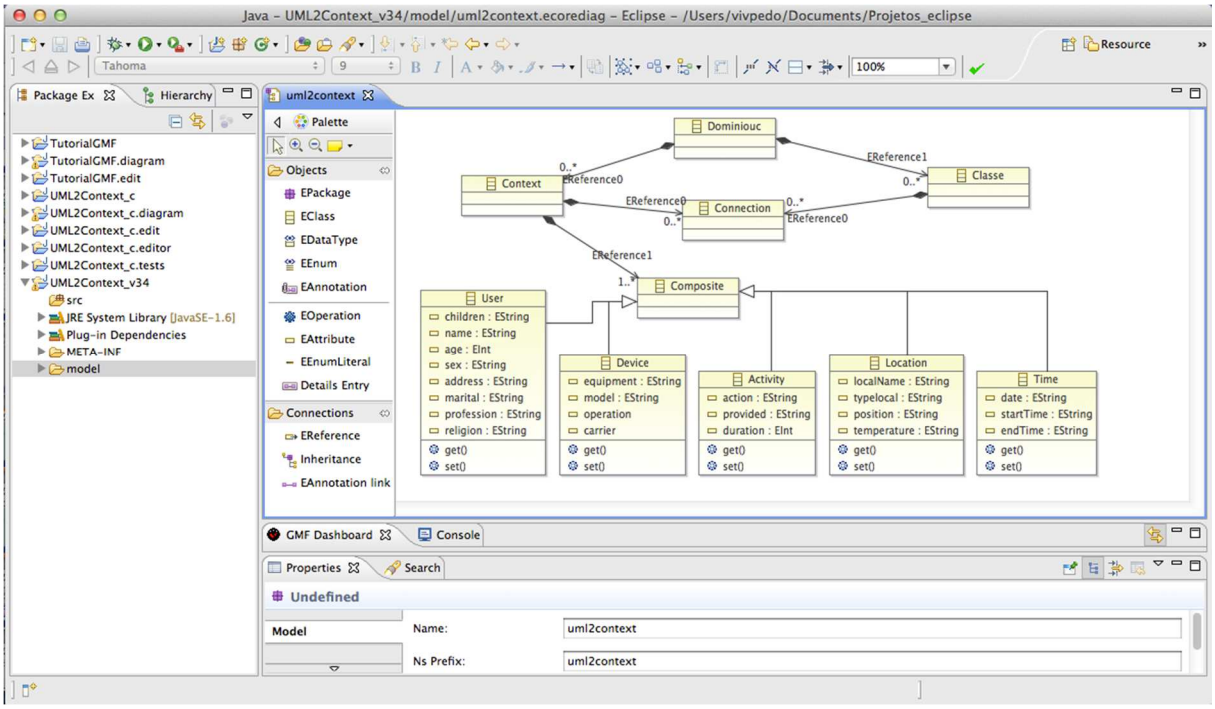
5.2 Processo de Desenvolvimento da Ferramenta

Neste ponto, o metamodelo de contexto criado guiou todo o processo de desenvolvimento, até chegar ao objetivo final que foi o ambiente de modelagem *UML2Context*. Conforme Farias et al. (2009), é necessário executar cinco etapas para o desenvolvimento da ferramenta de modelagem nesse nível. Ao instalar os plugins descritos na seção anterior deve ser marcado a opção *"Show dashboard view for the created project"* que disponibiliza um componente chamado *dashboard*, Figura 22, onde efetua-se os passos para a criação da ferramenta fazendo as configurações necessárias a cada passo.

Primeiramente então cria-se um arquivo Ecore com metamodelo definido. Para isso foi pego o arquivo padrão Ecore como modelo, e pelo eclipse e incluímos as metaclasses com suas responsabilidade, atributos, operações e relacionamentos. Deixamos definido as responsabilidades nas Notas, os atributos com o *EAttibute* do tipo *EString* (adaptação que o ecore faz da classe *java.lang.String*), os relacionamentos com o *EReference*, e definimos a cardinalidade da relação nos campos *Lower Bound* e *Upper Bound*.

Assim cria-se o arquivo "UML2Context.ecore" e então faz-se a importação do mesmo através do Dashboard para nossa aplicação no Eclipse. E através do comando "*Initialize ecore_diagram diagram file*" se pode visualizar o modelo que contém o arquivo Ecore como pode ser visto na Figura 21.

Figura 21: Modelo visual do metamodelo Ecore definido.



Fonte: Elaborado pela autora.

Na Figura 22 pode-se ver o *dashboard* numerado de acordo com a execução dos passos, que são descritos a seguir de forma resumida e que segue como descrito por Farias et al. (2009):

Etapa 1) Definição do modelo de domínio: nessa etapa o metamodelo proposto é especificado usando o EMOF. O componente Genmodel gera classes java de acordo com o meta-modelo. Onde se faz a importação do arquivo Ecore criado;

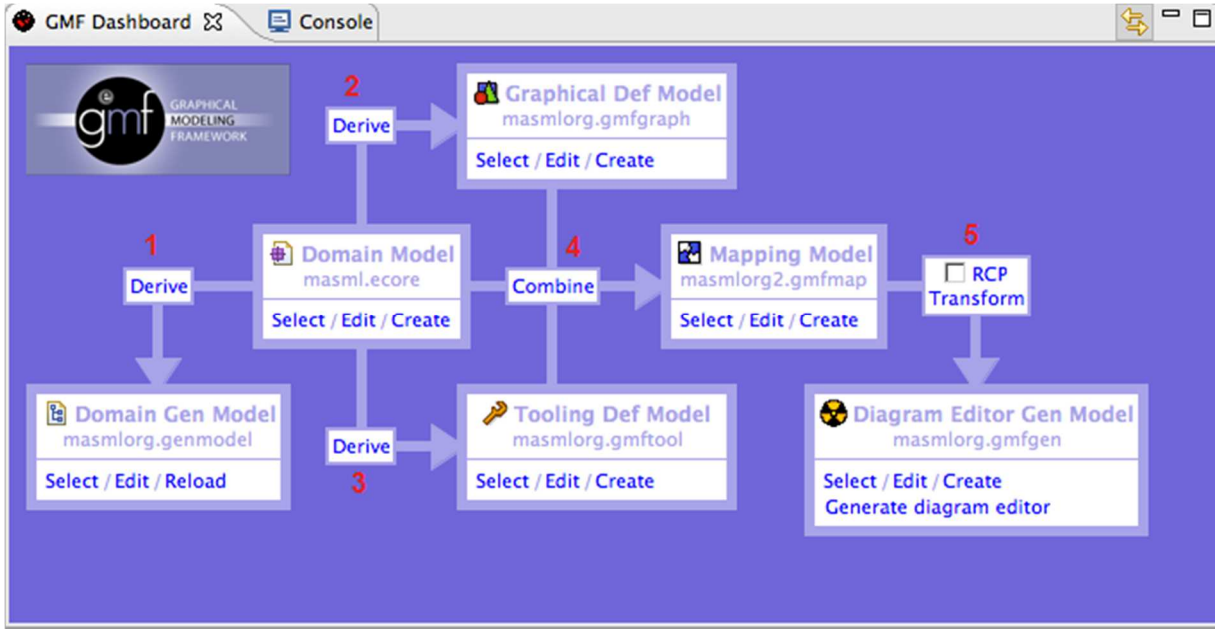
Etapa 2) Definição do modelo gráfico: nessa etapa é definida graficamente como será gerado nosso diagrama. É feita a definição das entidades e suas propriedades como também os relacionamentos entre elas (que foram especificados no modelo) que devem poder ser expressos na ferramenta de modelagem;

Etapa 3) Definição do modelo de ferramenta: nessa etapa, a partir do modelo de domínio e do modelo gráfico anteriores, é especificado quais elementos farão parte da paleta das ferramentas do nosso ambiente de modelagem e a criação da mesma;

Etapa 4) Definição do mapeamento: nessa etapa é feita construção de mapeamento entre os três modelos anteriores. Gera o mapeamento do diagrama e também um conjunto de referências, referenciando o meta-modelo ecore, o arquivo gmfgraph e gmftool. Faz-se o mapeamento de qual classe se associa a um componente gráfico e qual componente gráfico se associa a um elemento da paleta;

Etapa 5) Geração da ferramenta: nessa última etapa é feita a geração do código da ferramenta a partir do modelo de plataforma gerado na etapa anterior. Aqui que se utiliza o *plug-in* GMF para gerar a ferramenta.

Figura 22: Dashboard de criação.

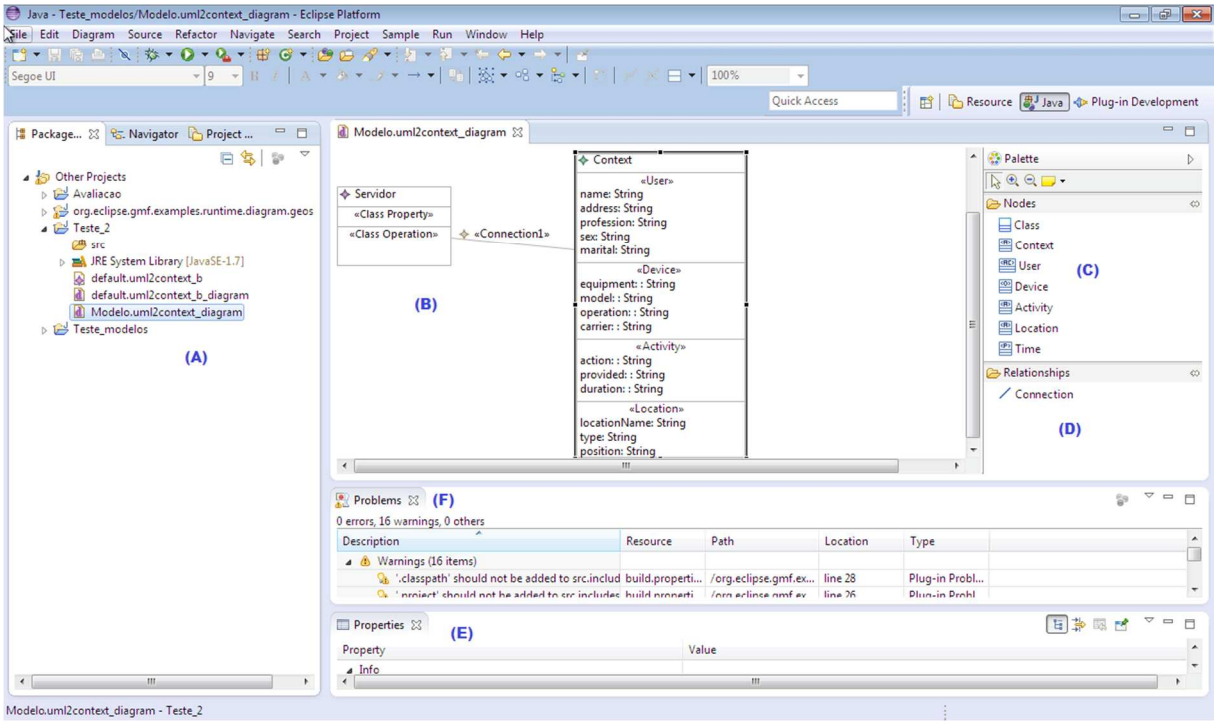


Fonte: Alterado pela autora (ECLIPSE, 2015).

5.3 Visão Geral da Ferramenta

Ao efetuar a geração da ferramenta são criados vários componentes onde cada um vai desempenhar uma funcionalidade específica. Como exemplo se pode ver na Figura 23 a ferramenta desenvolvida indicando onde se localiza cada componente criado.

Figura 23: Ferramenta UML2Context Tool criada nas etapas mostradas



Fonte: Elaborado pela autora.

Os componentes são criados pelo *plug-in* ao efetuar as etapas listadas anteriormente e onde cada um fica localizado quando o ambiente está pronto para uso. Na figura 23 mostramos a imagem da ferramenta UML2Context Tool e seus componentes localizados por letras e que estão resumidos abaixo:

a) ***Package Explorer***: tem o objetivo de permitir a organização de arquivos em uma estrutura de árvore para que seja possível um melhor gerenciamento e manipulação destes;

b) ***Modeling View***: tem o objetivo de permitir aos desenvolvedores visualizar e editar os modelos de forma interativa. Pois os modelos que são criados precisam ser visualizados para atender dois requisitos básicos de modelos que são compreensibilidade e a comunicação;

c) ***Nodes Palette***: tem o objetivo de permitir aos desenvolvedores criar instâncias de construtores, que podem fazer parte do diagrama de classes proposto pelo modelo;

d) ***Relationship Palette***: tem o objetivo de permitir a visualização dos relacionamentos dos construtores, que são visualizados na Modeling View e são instanciados na Nodes *Palette*;

e) ***Properties View***: tem o objetivo de permitir a manipulação de forma precisa das propriedades dos modelos. Ela vai exibir as propriedades definidas no metamodelo;

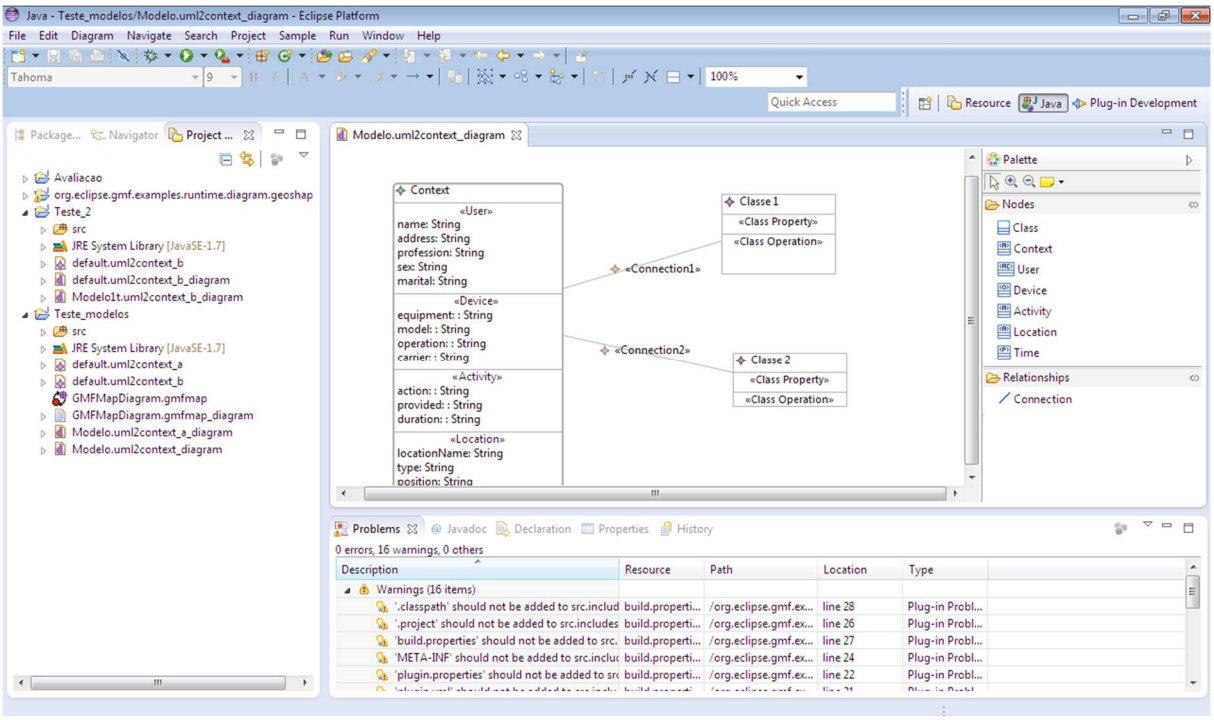
f) ***Problems View***: tem o objetivo de disponibilizar uma funcionalidade de "*validar modelo*", que valida os modelos criados em relação ao metamodelo da linguagem. Se existir alguma inconsistência ou item de atenção, ela será mostrada nesse componente. Assim é possível especificar se existe alguma inconsistência no modelo criado em relação à definição do metamodelo da linguagem.

Uma das preocupações constantes durante a elaboração do metamodelo de contexto juntamente com a extensão da UML consiste na identificação e representação das características dos conceitos presentes no paradigma de computação ciente de contexto dentro das metaclasses do contexto (são inseridas como propriedades das metaclasses).

5.4 Vantagens e Limitações da Ferramenta

Pode-se afirmar que, apesar da modelagem de contexto proposta através da linguagem já tenha proporcionado uma nova visão e entendimento do uso do contexto, sem uma ferramenta para utilizá-la dificultaria a aplicação dos conceitos criados. Como as aplicações atuais que utilizam UML não suportam todos os conceitos levantados foi fundamental que disponibilizássemos uma ferramenta para que modelagem de contexto com os termos propostos fosse efetivamente utilizada. Portanto a ferramenta de modelagem projetada foi fundamental para incorporar os elementos da linguagem de modelagem de contexto para uso em ambiente de desenvolvimento de forma fácil.

Figura 24: UML2Context Tool e as ações efetuadas



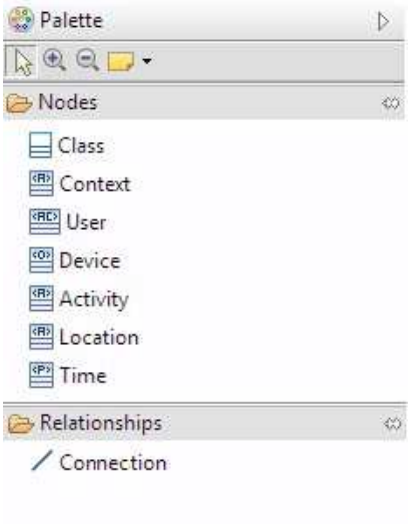
Fonte: Elaborado pela Autora.

A dificuldade na representação dos elementos da linguagem é um dos motivos para o uso de uma ferramenta de modelagem, trabalhando a questão da comunicação através da representação dos aspectos estáticos e dinâmicos das aplicações cientes de contexto. A ferramenta possibilita de forma fácil e rápida, através das paletas de trabalho a criação gráfica do contexto que será utilizado no software ciente de contexto. Pode-se ver na Figura 24 uma descrição simples de como foram criados os elementos na ferramenta para início de uma modelagem.

No *Problems View*, componente do GMF é disponibilizada a validação dos modelos criados em relação ao metamodelo da linguagem e sinaliza se existir alguma inconsistência ou ponto de atenção a ser verificado. Também o UML2Context Tool foi projetado de forma a ser estendido para incorporar os demais modelos gráficos estáticos da linguagem. Assim outros gráficos UML podem ser inseridos de forma mais fácil utilizando o mesmo metamodelo criado no arquivo ecore, e com as ferramentas que já existem.

Na paleta de ferramentas Figura 25, foram disponibilizados os elementos "*Context*", que já é inserida com repartições por elementos, os elementos individualizados para que possam ser inseridos dentro do elemento *context*. E uma classe comum com propriedades e métodos que faz o relacionamento com o contexto.

Figura 25: Paleta de ferramentas do UML2Context



Fonte: Elaborado pela Autora.

Finalizando o capítulo comenta-se algumas das limitações da ferramenta que podem ser sanadas à medida que se siga explorando o desenvolvimento da mesma:

Uma questão é que apesar do UML2Context Tool ter sido projetado de forma a ser estendido para incorporar os demais modelos estáticos da linguagem, hoje uma das limitações atuais da ferramenta é que sua versão atual apenas contempla o diagrama de classes. No entanto como descrito anteriormente outros diagramas podem ser incluídos.

E também a ferramenta ainda não realiza geração de código, funcionalidade já provida por algumas ferramentas de modelagem existentes como o ERWin Data Modeler[5] e PowerDesign[6] que exportam scripts para vários tipos de bancos. Nessa primeira versão da UML2Context Tool, o foco foi desenvolver a ferramenta com uma arquitetura extensível para a incorporação de outros modelos e a captura do metamodelo UML2Context elaborado.

[5] ERWin: http://erwin.com/products/data-modeler
[6] Power Design: http://www.software.com.br/p/powerdesigner

6 AVALIAÇÃO

Este trabalho propõe uma linguagem para modelagem de contexto com o objetivo de apoiar o desenvolvimento de sistemas cientes de contexto, como previamente já mencionado. Este capítulo foca, então, em avaliar a linguagem proposta. Para isso, um estudo experimental foi projetado e executado seguindo boas práticas amplamente reconhecidas na literatura, as quais são definidas em (WOHLIN, 2012).

Na seção 6.1 esclarece a metodologia utilizada para avaliação. Na seção 6.2 aborda-se como foi efetuado o experimento controlado. Na seção 6.3 é explicado como foram desenvolvidas as questões e quais foram os softwares escolhidos para elaboração da avaliação. Os resultados obtidos são discutidos na seção 6.4 apoiados por testes estatísticos e análise qualitativa.

6.1 Metodologia do Estudo

A seção 6.1.1 esclarece a metodologia utilizada para avaliação bem como, o objetivo e as questões de pesquisa abordados; Na seção 6.1.2 explica como foram formuladas as hipóteses para avaliação. E por fim na seção 6.1.3 detalha as variáveis de quantificação utilizadas para os cálculos efetuados.

6.1.1 Objetivo e Questões de Pesquisa

Foram utilizados para comparação, modelos de UML pura e modelos na proposta da linguagem UML2Context, em um contexto particular: o uso e compreensão necessários pelos desenvolvedores para entender os modelos e para produzir a implementação correspondente. A seguir, o objetivo do estudo experimental é definido seguindo o template GQM - Goal, Questions, Metrics (WOHLIN, 2000):

Analisar as abordagens UML2Context e UML pura

com o propósito de investigar o impacto

com relação ao esforço de interpretação, corretude e má interpretação

a partir da perspectiva do desenvolvedor

no contexto de desenvolvimento de software.

Com base neste objetivo, três Questões de Pesquisa (QP) são formuladas:

- **QP1:** A UML2Context afeta a eficiência de desenvolvedores para identificar a melhor forma de implementação do contexto?
- **QP2:** A UML2Context influencia nos esforços investidos pelos desenvolvedores para implementar corretamente o modelo de contexto?
- **QP3:** A UML2Context reduz a taxa de má interpretação na modelagem de contexto em comparação com a modelagem em UML pura?

É importante destacar que a seleção do domínio do estudo é representativa de situações em que os desenvolvedores implementem classes e/ou elementos, com base nos softwares cientes de contexto escolhidos e que são abordados na seção 6.3. Baseado nestas QPs, três hipóteses foram estabelecidas, as quais são apresentadas na próxima seção.

6.1.2 Formulação de Hipóteses

A análise dos resultados será feita sobre três hipóteses.

Primeira hipótese (H1): *A primeira questão de pesquisa investiga se os indivíduos por meio dos modelos de contexto com UML2Context produzem uma taxa de respostas corretas inferior ou superior do que a modelagem com a UML pura*. Assim, espera-se que os modelos UML2Context possam ajudar os desenvolvedores a mudar mais rapidamente entre os pontos de vista de comportamento e estruturais quando visualizarem a modelagem do contexto. As hipóteses avaliam a TRC (Taxa de Respostas Corretas). Essas hipóteses são resumidas como segue:

- **Hipótese Nula 1 ($H_{1\text{-}0}$):** A taxa de identificação da implementação correta de contexto na UML2Context é igual (ou maior) do que na UML pura.
  - H1-0: TRC (UML2Context) ≥ TRC (UML pura)
- **Hipótese Alternativa 1 ($H_{1\text{-}1}$):** A taxa de respotas corretas da implementação na UML2Context é menor do que na modelagem UML pura.
  - H1-1: TRC (UML2Context) < TRC (UML pura)

Segunda hipótese (H2): *A segunda questão de pesquisa investiga se os desenvolvedores investem menos ou mais esforço para detectar a implementação correta nos modelos UML2Context do que nos modelos UML pura.* A abordagem do contexto de maneira própria pode ajudar os desenvolvedores na análise. Assim, a expectativa é que quanto melhor for abordado o contexto no modelo, menor será o esforço dos desenvolvedores para chegar a uma conclusão e consequêntemente em uma resposta. Auxiliando assim no trabalho de desenvolvimento de sistemas cientes de contexto, pois as dificuldades de intepretação do contexto podem desacelerar o trabalho de desenvolvimento, uma vez que os desenvolvedores de frente para a complexidade dos requisitos e modelos, evitam fazer qualquer implementação (FARIAS, 2012). Ou seja, busca-se saber se usando a UML2Context os desenvolvedores tendem a ficar mais produtivos. Isto leva à segunda hipótese nula e uma hipótese alternativa como se segue onde é comparado o EI (Esforço de interpretação):

- **Hipótese Nula 2 ($H_{2\text{-}0}$):** O esforço para identificar a implementação correta na UML2Context é igual (ou maior) do que em UML pura.
  - **$H_{2\text{-}0}$:** EI (UML2Context) ≥ EI (UML pura)
- **Hipótese Alternativa 2 ($H_{2\text{-}1}$):** O esforço para identificar a implementação correta na UML2Context é menor do que na UML pura.
  - **$H_{2\text{-}1}$:** EI (UML2Context) < EI (UML pura)

Terceira hipótese (H3): *A terceira pergunta de pesquisa investiga se a taxa de má interpretação dos modelos de contexto (TMI) por parte dos desenvolvedores é superior ou inferior na UML2Context do que na UML pura*. É difícil para os desenvolvedores pensarem da mesma forma e, portanto, a produção de código com a mesma semântica. A principal razão é que a implementação de software amplamente depende de fatores cognitivos (LANGE e CHAUDRON, 2004). Alguém poderia considerar que os conceitos da UML2Context ou da UML pura pode interferir negativamente em um entendimento comum de modelos de contexto por diferentes desenvolvedores. Algumas vezes os desenvolvedores podem precisar

entender exatamente o significado real das relações entre os elementos do contexto (além de todas as outras relações) na documentação quando elas, na verdade, podem ser estabelecidas durante a execução do sistema. Como os desenvolvedores precisam examinar todos os elementos contidos em um contexto, suas análises extras podem aumentar as oportunidades de interpretações divergentes. No entanto essa expectativa não pôde ocorrer se a modelagem na UML2Context facilitar esta visão para estes profissionais. Isso levaria a as seguintes hipóteses nula e alternativa:

- **Hipótese Nula 3 (H_{3-0}):** A taxa de erros de interpretação na UML2Context é igual ou maior do que na UML pura.
  - **H_{3-0}:** TMI (UML2Context) ≥ TMI (UML pura)
- **Hipótese Alternativa 3 (H_{3-1}):** A taxa de má interpretação na UML2Context é menor do que na UML pura.
  - **H_{3-1}:** TMI (UML2Context) < TMI (UML pura)

6.1.3 Variáveis e Métodos de Quantificação

A variável independente desta avaliação é a escolha da linguagem de modelagem. É nominal e dois valores pode ser assumidos: modelagem de contexto em UML2Context e modelagem de contexto em UML pura. Essas variáveis descrevem as questões de avaliação, e objetiva a investigação dos seus impactos sobre as seguintes variáveis dependentes (WOHLIN, 2012).

A primeira variável dependente é a taxa de identificação da melhor implementação para a modelagem do contexto que chamaremos de Taxa de Respostas Corretas (TRC). A variável "TRC" é concebida para medir o índice global de identificação da melhor implementação para a modelagem do contexto detectadas pelos participantes (Questão1). Ela representa a razão entre o número de indivíduos que identificam a implementação correta em uma questão, dividida pelo número total de indivíduos, ou seja, TRC = <número de repostas corretas> / <número de participantes>.

A segunda variável dependente é o esforço para a identificação da implementação correta para modelagem do contexto. A variável esforço será a "EI" que representa o Esforço de Identificação pela a média de tempo (minutos) gastos pelos participantes para detectar inconsistências em uma pergunta (Questão 2). Os indivíduos que não identificam a questão correta no modelo indica explicitamente que são incapazes de alcançar uma implementação adequada a partir dos diagramas disponíveis.

A terceira variável dependente é a Taxa de Má Interpretação (TMI) dos modelos. Esta variável representa o grau de variação das respostas (Questão 3). Ou seja, mede a concentração das respostas ao longo das cinco alternativas possíveis nas questões, onde a última alternativa para marcação é de que não foi possível indetificar a implementação correta por problemas no modelo. Nossa preocupação é se os modelos ficaram complexos a ponto do indivíduo não conseguir localizar a alternativa correta. Uma má interpretação dos modelos impedindo o indivíduo de identificar a implementação correta não é necessariamente problemática (LANGE e CHAUDRON, 2006) se todos os indivíduos têm a mesma interpretação para uma pergunta, veremos então que apenas o modelo não cumpriu seu objetivo de facilitar o trabalho do desenvolvimento. Nesse caso conclui-se que as más interpretações dos diagramas não levam a erros de interpretação (TMI = 1). No entanto, se as respostas dos desenvolvedores se distribuírem aleatóriamente sem padrão ao longo das cinco

alternativas, então os modelos causam interpretações erradas graves (TMI = 0). Ou seja, a taxa de erros de interpretação é 0 se respostas são distribuídos igualmente sobre todas as opções. E a taxa será 1 se as respostas estão concentradas em apenas uma opção. Segundo Lange e Chaudron (2006) esta variável pode ser medida como se segue:

$$\boldsymbol{TMI}\,(k_0, \ldots, k_{K-1}) = 1 - 2\,\frac{\sum_{0 \le i < K} k_i i}{N(K-1)}$$

Onde....

K: número de alternativas para a questão

ki: O número de vezes alternativa i foi selecionado, em que 0 ≤ i <K e (para todos i: 0 ≤ i <K - 1: ki ≥ + 1)

N: A soma das respostas sobre todas as alternativas: $N = \sum_{0 \le i < K} k_i$

Simplificando podemos dizer o TMI, possui uma unidade simples e calcula o número de vezes que a alternativa “i” foi selecionado (entre as cinco alternativas), dividindo pela soma das respostas sobre todas as alternativas.

6.2 Experimento Controlado

Foram elaborados dois questionários Anexo I, utilizando sistemas cientes de contexto, e foram desenvolvidas questões utilizando a linguagem de modelagem UML pura e a linguagem de modelagem desenvolvida neste trabalho, a UML2Context. Assim foi efetuado um experimento controlado com o objetivo de investigar o impacto da UML2Contex nas seguintes variáveis:

- Taxa de acertos por modelagens;
- Esforço dos desenvolvedores para identificar a implementação correta;
- Taxa de má interpretação dos desenvolvedores.

O experimento foi realizado com profissionais que atuam no mercado de trabalho atual em diferentes empresas privadas de desenvolvimento de software e que possuem conhecimento e experiência em desenvolvimento e/ou análise. E também com alunos de graduação e mestrado com conhecimento em programação e análise. Para avaliação, 24 indivíduos foram envolvidos, onde dispondo dos diagramas de classe e de sequência eles deveriam assinalar a melhor resposta sobre a implementação baseado no entendimento dos diagramas. Os indivíduos foram selecionados com base em dois critérios fundamentais: (1) profissionais que atuam no mercado atual de desenvolvimento de software utilizando boas práticas e ter experiência prática relacionada a desenvolvimento de software e com algum conhecimento em modelagem de software; e (2) alunos que cursam atualmente graduação ou mestrado com conhecimento de programação e análise. Os indivíduos escolhidos agregam um valor elevado em princípios fundamentais de modelagem de software e programação, uma vez que contam com experiência ou conhecimento, informações atualizadas sobre o mercado ou acadêmico e tentam aplicar boas práticas no desenvolvimento.

Todos os indivíduos foram submetidos a duas modelagens de contexto (UML2Context e modelagem com UML pura) para permitir a comparação dos pares combinados de material experimental. Cada modelagem teve um questionário com cinco questões de múltipla escolha.

Um questionário teve apenas perguntas com modelos em UML pura enquanto o outro teve apenas perguntas com modelos UML2Context. Os participantes foram aleatoriamente divididos e igualmente distribuído a estes questionários. Assim a avaliação experimental deste estudo é, por definição, uma avaliação que pode ser chamado de balanceada.

Os questionários foram elaborados de forma que contivesse as duas modelagens prontas e o desenvolvedor pudesse observá-las e avaliar as implementações que poderiam ser feitas baseando-se no entendimento dos digramas. Primeiramente foram escolhidos cinco sistemas que utilizam ciência de contexto, que fossem de segmentos diferentes e que abordassem elementos diferenciados no contexto. Posteriormente analisou-se como os sistemas abordavam o contexto e foram feitas duas modelagens: uma com a linguagem UML2Context e outra com a linguagem UML pura. Por fim, foram elaboradas questões sobre a implementação dos softwares.

Ao elaborar os questionários com as duas modelagens, houve a preocupação de que os participantes ganhariam informação com a primeira modelagem e ao realizar a análise da segunda já contariam com soluções hipotéticas. Para minimizar o "*ganho de informação*" com a primeira análise algumas estratégias experimentais (KITCHENHAM, 2008) foram seguidas. Primeiro os modelos utilizados no estudo foram fragmentos de diagramas de classe e diagramas de sequência, baseados nos modelos que foram utilizados para desenvolver as aplicações reais. Os participantes não tinha nenhuma informação prévia e sem o conhecimento acumulado sobre a semântica dos elementos dos modelos. Em segundo lugar, cada pergunta tinha um diagrama de classe e sequência representando diferentes funcionalidades de um software. Em terceiro lugar, cada par de modelos estruturais e comportamentais tiveram diferentes tipos de modelagens onde os elementos se comportavam completamente diferentes. Portanto, podemos supor que o desempenho dos participantes não foi influenciado pelo primeiro questionário respondido.

Nas duas modelagens os indivíduos receberam um diagrama de classe correspondente (estrutural) e um diagramas de sequência (comportamentais), cada questão tinha o nome da aplicação ciente de contexto e na orientação da resposta uma breve idicação do segmento da aplicação (saúde, financeira, pagamento, turismo e acessibilidade). Os participantes foram questionados sobre como fariam a implementação de uma determindada classe com base nestes diagramas. Ou seja, em vez de estimulados a rever ou inspecionar os diagramas, os participantes foram encorajados a implementar determinados elementos do modelo com o objetivo de identificar como os desenvolvedores lidariam com os elementos do contexto que executavam de tarefas concretas de engenharia de software. A tarefa consistia em escolher a implementação mais apropriados entre as cinco opções de resposta, mas todas as questões tiveram uma quinta opção em que poderiam indicar que foi detectada uma inconsistência nos modelos. E também foram estimulados a justificar as suas respostas na folha de respostas. Em cada questionário os participantes foram obrigados a registrar o tempo investido para responder cada modelagem. No total, dez perguntas foram respondidas. Mais detalhes da concepção experimental pode ser encontrada em Farias (2012).

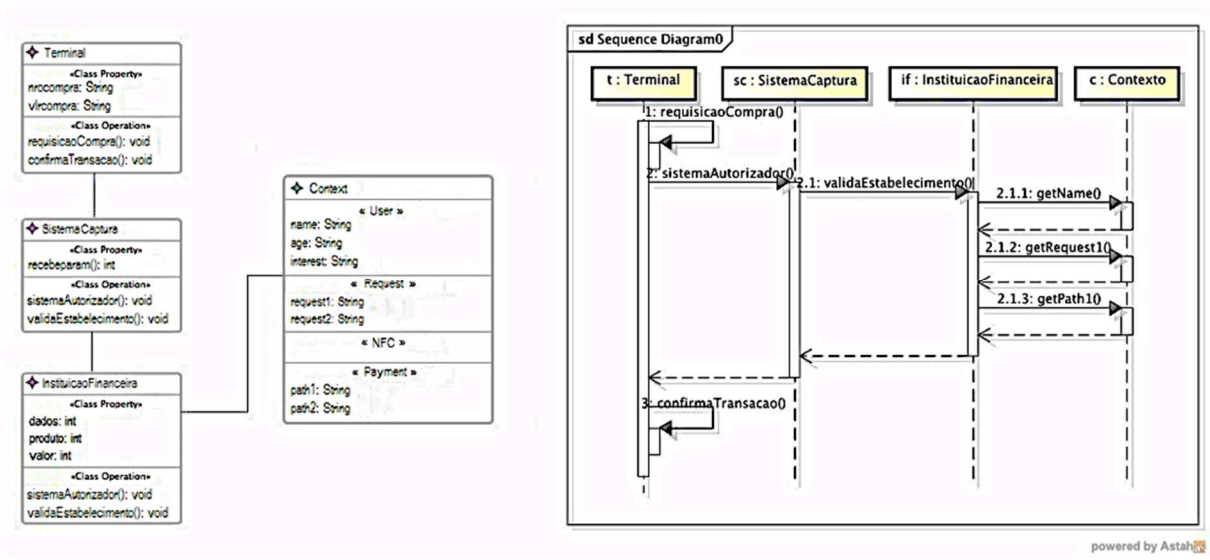
Com os resultados buscou-se a perspectiva dos desenvolvedores, se a extensão proposta melhora a representação e compreensão de contexto e os auxiliou ou facilitou na escolha da implementação adequada ou na identificação de inconsistências nos modelos.

6.3 Questionário

O questionário desenvolvido possui 5 questões utilizando a UML2Contetx e 5 questões utilizando a UML pura. Nas questões utilizando a linguagem UML2Context o diagrama de classe foi elaborado na ferramenta da linguagem expondo objetivamente os elementos do contexto. A UML2Context, foi baseada na UML mas com a aplicação de técnicas de extensão da linguagem para que pudesse contemplar os elementos de contexto. A utilização da a UML estendida foi feita para que fosse de fácil uso por indivíduos com baixa qualificação (ou experiência) e indivíduos altamente qualificados (ou experientes). Assim não é necessário nenhum treinamento ou esclarecimento anterior para os usuários que vão visualizar os diagramas. Nas questões utilizando a UML pura os diagramas de classe foram elaborados na ferramenta Atash (ASTAH, 2014), pois a mesma é conhecida no meio profissional facilitando o entendimento dos diagramas. Na UML pura foram empregados conceitos básicos e elementos de modelagem orientada a objetos clássicos, e como já foi mencionado, a UML é o padrão para a concepção de sistemas de software (*UML, 2014*), facilitando assim a compreensão por parte dos desenvolvedores. Nos dois casos foram elaborados diagramas de sequência para exemplificar a utilização das classes mostradas e os mesmos foram elaborados na ferramenta Astah.

A Figura 27 apresenta um exemplo ilustrativo dos modelos utilizados em nosso estudo: uma classe e um diagrama de sequência da linguagem UML2Context desenvolvida neste trabalho. A notação apoia a representação visual de aspectos, relações entre os elementos e outros conceitos da UML2Context. O diagrama de classe foi feito utilizando o UML2Context onde os estereótipos como << User >> representam um elemento dentro do contexto.

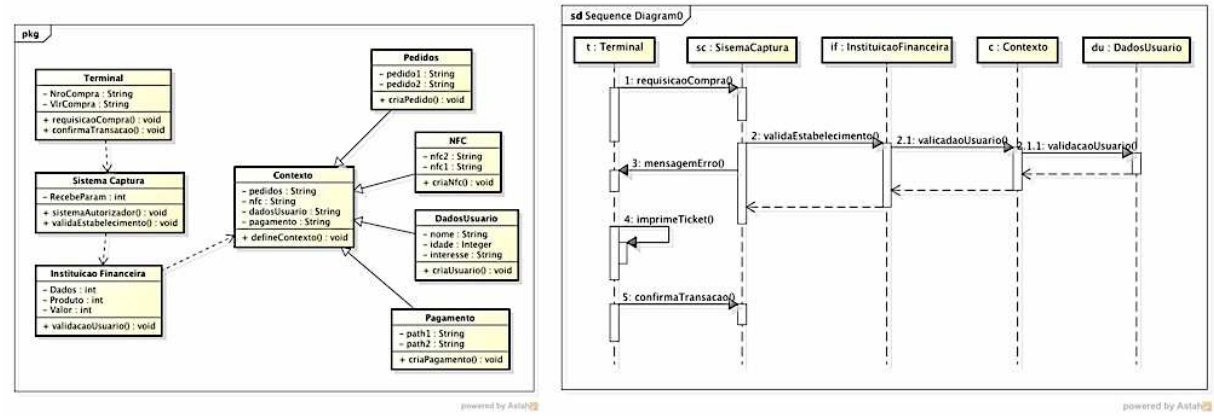
Figura 26: Modelo de diagramas das questões de UML2Context.



Fonte: Elaborado pela Autora.

Na Figura 28, é apresentado um exemplo ilustrativo dos modelos utilizados para representar a UML pura. Tanto os diagramas de classes como os de sequência foram feitos na ferramenta Astah. Os diagramas de classes foram elaborados a partir dos trabalhos de dissertação dos sistemas cientes de contexto escolhidos. Para manter um padrão nas questões os diagramas foram reconstruídos a partir de diagramas já existentes e informações constantes nos trabalhos, as informações ausentes foram adaptadas.

Figura 27: Modelo de diagramas das questões de UML pura.



Fonte: Elaborado pela Autora.

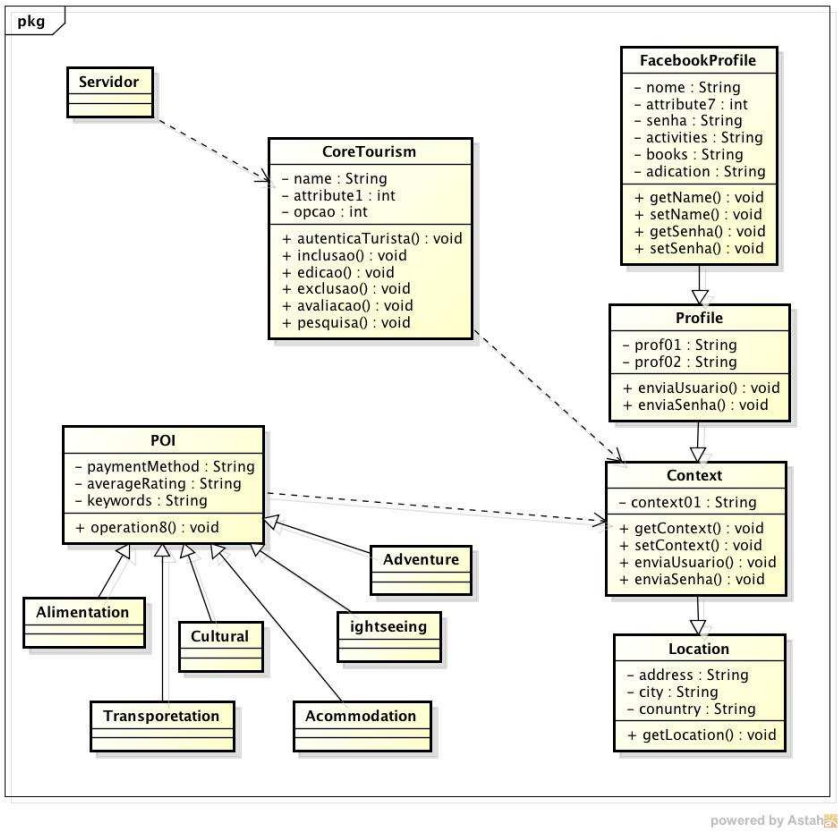
As Figuras 27 e 28 representam os diagramas estruturais utilizados, elas foram aplicadas sobre o mesmo software ciente de contexto de maneiras diferenciadas para que possamos utilizar as abordagens e a maneira como os desenvolvedores interpreta as mesmas para avaliação. Para elaboração das questões de avaliação foram escolhidos cinco softwares que trabalhassem com ciência de contexto e onde o contexto fosse o principal aspecto da aplicação. A seguir os softwares resumindo suas funções.

6.3.1 Software UbiTour

O UbiTour (COSTA, 2013) é uma dissertação de mestrado, defendida em 2013 no Programa de Pós-Graduação em Computação Aplicada (PIPCA) da Unisinos, na área de Turismo Ubíquo. O modelo emprega diversos conceitos da computação ubíqua, como ciência de contexto, transparência ao usuário, serviços baseados em localização, perfis, restrições, para que o turista possa, entre outras atividades, consultar quais são os pontos de interesse turísticos mais indicados para o seu perfil em um determinado contexto (COSTA, 2013).

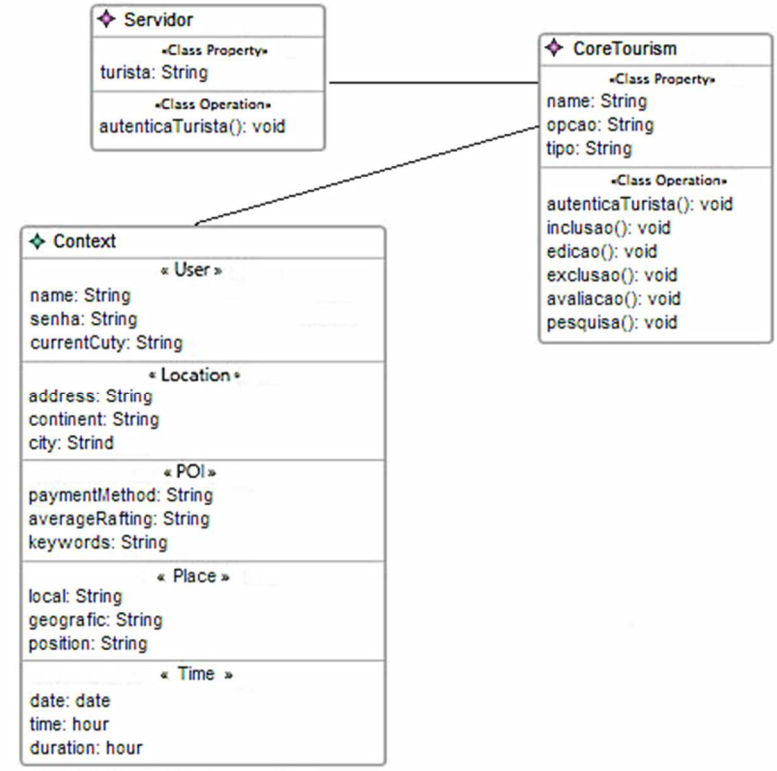
No software existem várias classes para a implementação do sistema, mas umas das principais é a classe *Context* que representa um contexto na aplicação que pode ser um perfil do turista (classe *Profile*), Localização (*Location*) ou pode estar contida em um Ponto de Interesse (POI), Rota (*Route*) ou Evento (*Event*) (COSTA, 2013). Na Figura 29 pode-se observar as classes principais da aplicação que foram modeladas com a UML padrão, exibindo um modelo de diagrama completo das classes utilizadas na aplicação. Na Figura 30 foi feita uma abstração do modelo e desenvolvido um diagrama no UML2Context onde pode-se observar o contexto modelado no UML2ContextTool.

Figura 28: Modelagem do UbiTour em UML pura



Fonte: Adaptado de Costa (2013)

Figura 29: Modelagem do UbiTour no UML2Context



Fonte: Elaborado pela Autora.

6.3.2 Software 4iPay

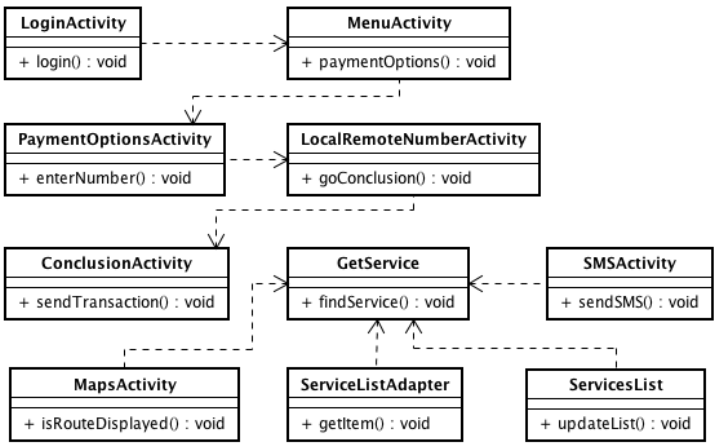
O 4iPay (ROEHRS, 2012) é uma dissertação de mestrado, defendida em 2012 no PIPCA da Unisinos, na área de Comércio Ubíquo. O trabalho propõe um modelo de pagamento móvel envolvendo diferentes formas de conexão de acordo com o contexto do usuário. No caso de transações de pagamento o usuário pode cadastrar preferências ou o sistema pode detectar automaticamente comportamentos desejados de acordo com o contexto em que ele se encontra com seu dispositivo (ROEHRS, 2012). Um cenário de pagamento móvel é composto pela rede de dados que um dispositivo móvel está usando e a possibilidade da mesma ser alterada automaticamente de acordo com as preferências do usuário e sua localização conforme as opções disponíveis de conexão (mostradas na Figura 31).

Figura 30: Conexões no 4iPay para dispositivos móveis

Fonte: Adaptado de Roehrs (2012).

A dissertação não produziu um diagrama abordando especificamente o contexto, entretanto é possível ter uma ideia da aplicação pela Figura 32 que apresenta o diagrama de classe do modelo do lado servidor. Nesse diagrama podem ser visualizadas as classes responsáveis pelo atendimento dos eventos de persistência e de recepção e encaminhamento dos serviços. Conforme o padrão de arquitetura DAO (Data *Access Object),* responsável pelos principais métodos de persistência da aplicação. Também mostra as principais classes de acesso da aplicação, (*LoginService, SaveService* e *ListService*), responsáveis pelo controle de acesso e autenticação, persistência das transações e por fim, a classe de consulta das transações, respectivamente nesta ordem (ROEHRS, 2012).

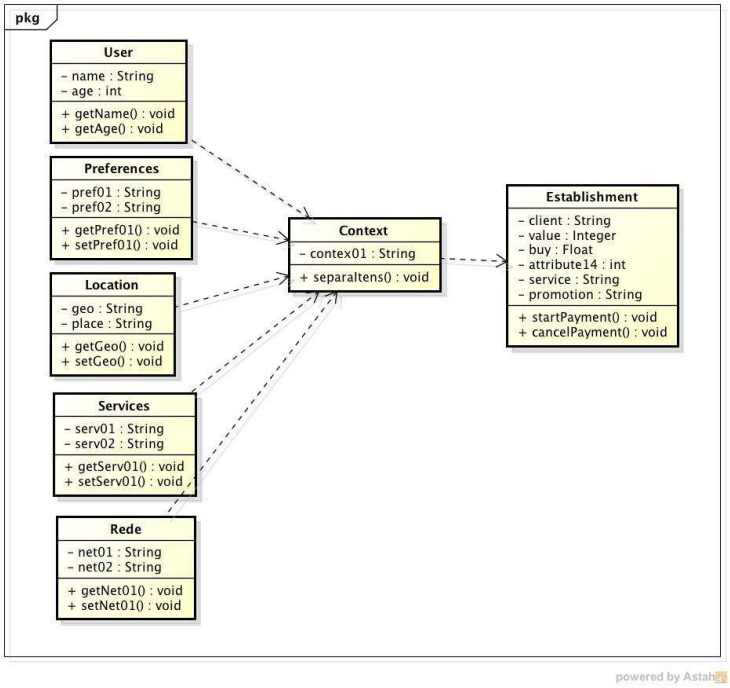
Figura 31: Diagrama de classe modelo servidor do 4iPay



Fonte: Roehrs (2012).

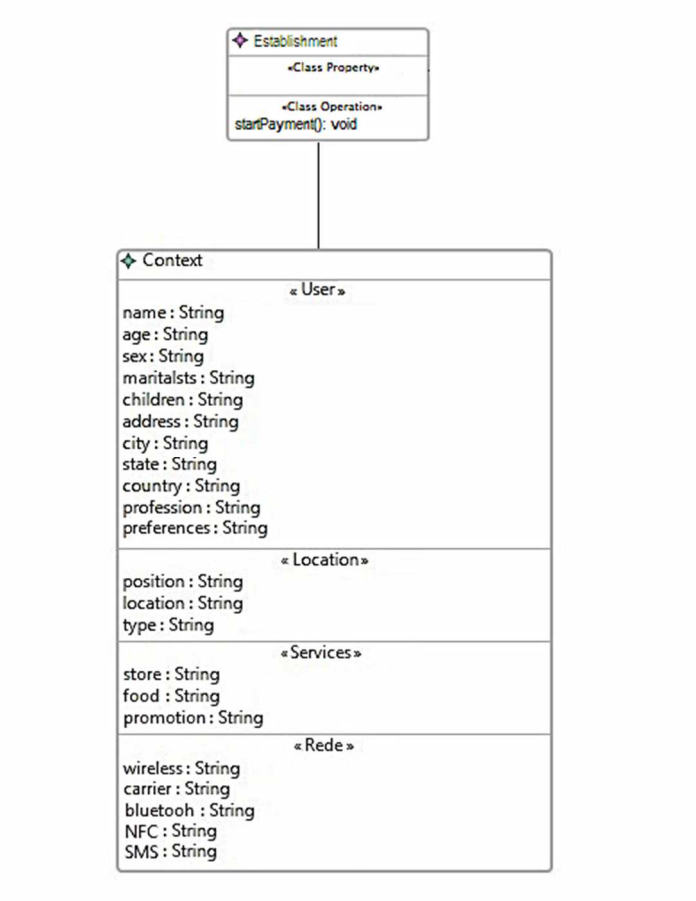
Na Figura 33 pode-se observar um diagrama de classe modelado com UML pura em Astah e que incluem os elementos utilizados na aplicação.

Figura 32: Modelagem do 4iPay em UML pura



Fonte: Elaborado pela autora.

Figura 33: Modelagem do 4iPay no UML2Context



Fonte: Elaborado pela autora.

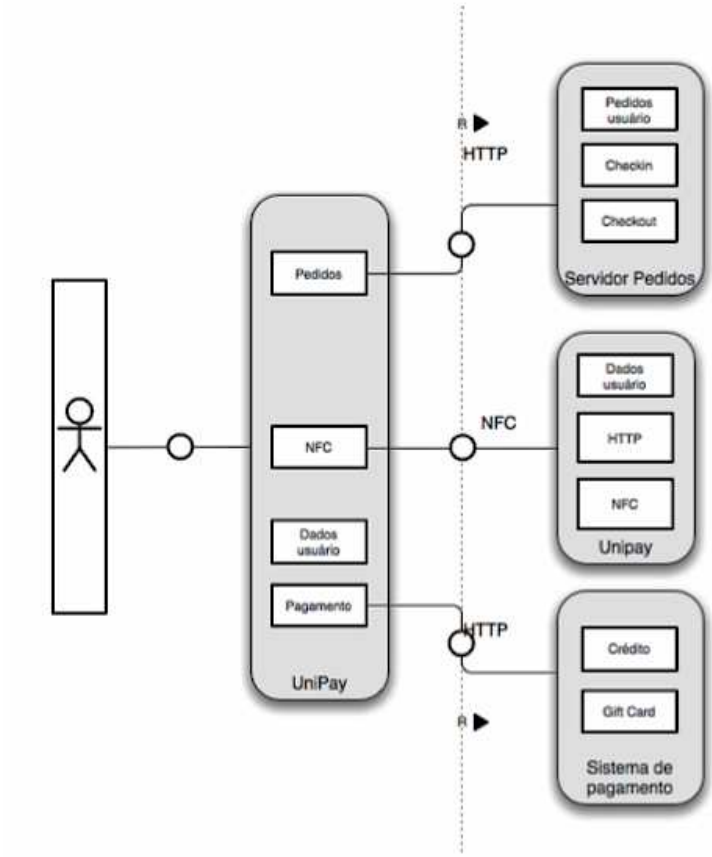
Complementarmente, na Figura 34 é possível observar o mesmo contexto modelado no UML2Context, onde os elementos são simplificados para apenas dois elementos, no entanto contendo todos os dados necessários para contemplar o contexto.

6.3.3 Software UniPay

O UniPay (JOST, 2012) é uma dissertação de mestrado, defendida em 2012 no PIPCA da Unisinos, na área de Pagamento Ubíquo. O modelo leva em consideração as informações de contexto do usuário e busca oferecer uma solução diferenciada, através da interface da aplicação com um servidor de pedidos no estabelecimento comercial (JOST, 2012).

A aplicação utiliza informações do contexto para realizar operações. Como exemplo, o usuário pode receber notificações de algum desconto em sua loja preferida caso encontre-se próximo a ela e receber informações dos estabelecimentos. Esta localização também pode ser utilizada para agregar segurança à transação financeira, garantindo que o usuário que está realizando a transação encontra-se de fato no estabelecimento. Utilizando a localização um dispositivo móvel informa sua posição através do uso do GPS, da triangulação das redes, da operadora de telefonia ou mesmo da rede Wi-Fi. A localização pode ser utilizada para oferecer serviços específicos, tendo como característica a utilização de preferências do usuário, para que as informações de contexto sejam repassadas de uma maneira direta, sem ser invasivo. Na Figura 35 pode-se ter uma visão geral do funcionamento do UniPay.

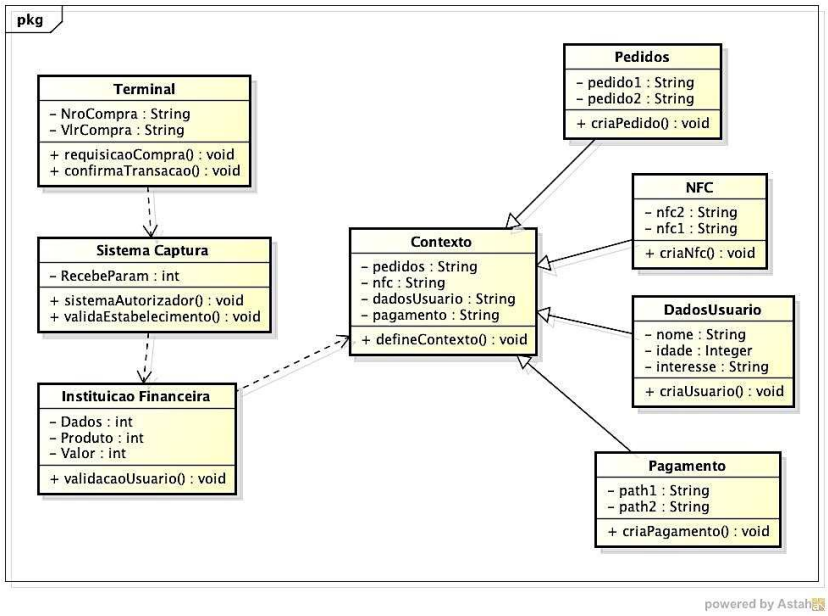
Figura 34: Visão Geral do modelo UniPay



Fonte: Jost (2012).

Na Figura 36 foi feito um diagrama de classe em UML Pura modelando o contexto de acordo com os elementos mencionados no trabalho usados na aplicação.

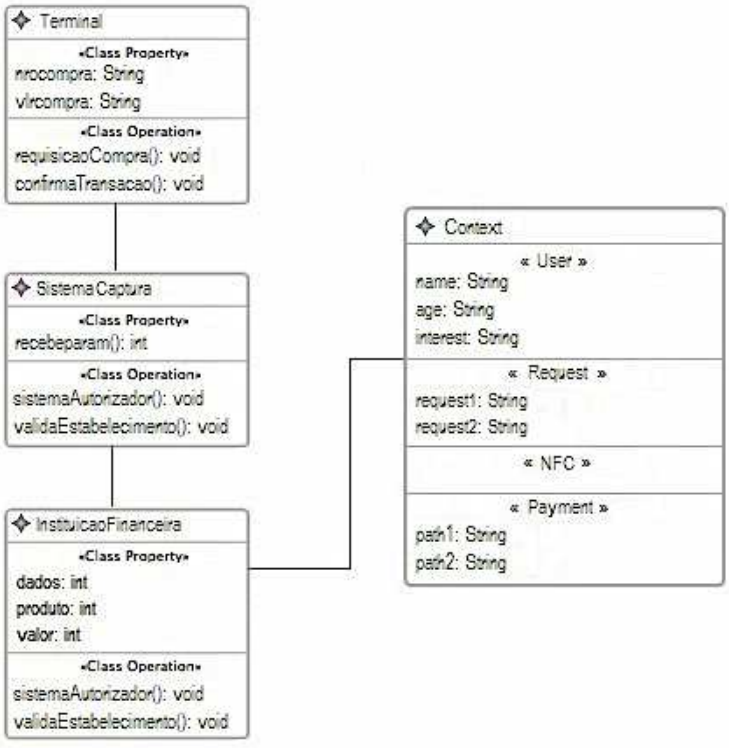
Figura 35: Modelagem do UniPay em UML pura



Fonte: Elaborado pela autora.

Por fim, na Figura 37 pode-se observar o contexto modelado no UML2Context, onde os elementos foram simplificados para o modelo do trabalho.

Figura 36: Modelagem do UniPay no UML2Context



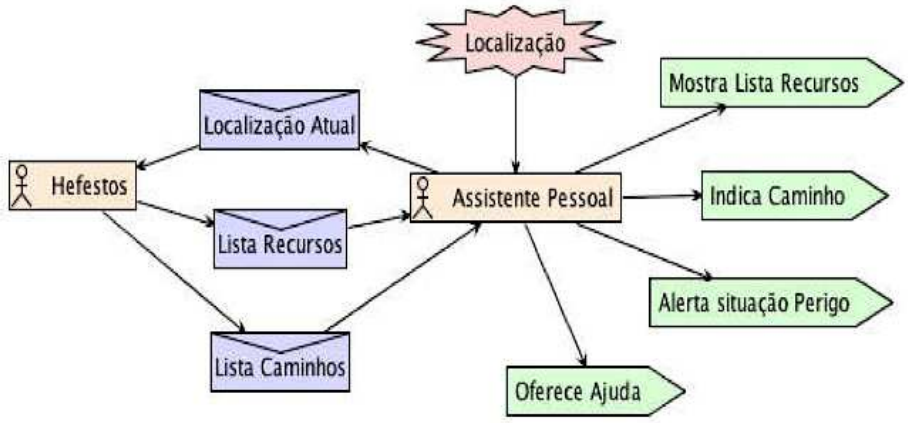
Fonte: Elaborado pela autora.

6.3.4 Software Hefestos

O Hefestos (TAVARES, 2012) é uma dissertação de mestrado, defendida em 2012 no PIPCA da Unisinos, na área de Acessibilidade Ubíqua. O modelo é orientado ao suporte de pessoas que possuam deficiência motora nas pernas. O sistema comporta perfis de usuários e recursos disponíveis nos contextos onde o sistema for usado, por exemplo uma universidade ou um clube de campo, para oferecer aos usuários com deficiência motora suporte à acessibilidade (TAVARES, 2012). Baseado no software Tirésias (FALK, 2013) que possui o mesmo conceito porém para apoio aos deficientes visuais. O Hefestos foi adaptado para identificar contexto para cadeirantes.

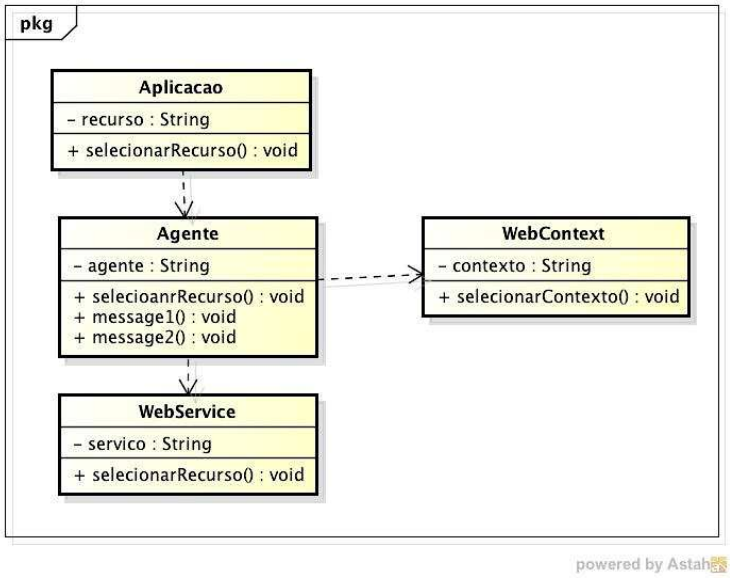
Na Figura 38 pode-se observar a modelagem do agente pessoal do Tirésias que atua concentrando o contexto. A Figura 39 apresenta o contexto modelado no Astah e na Figura 40 o mesmo contexto modelado no UML2Context.

Figura 37: Modelagem do agente pessoal da aplicação



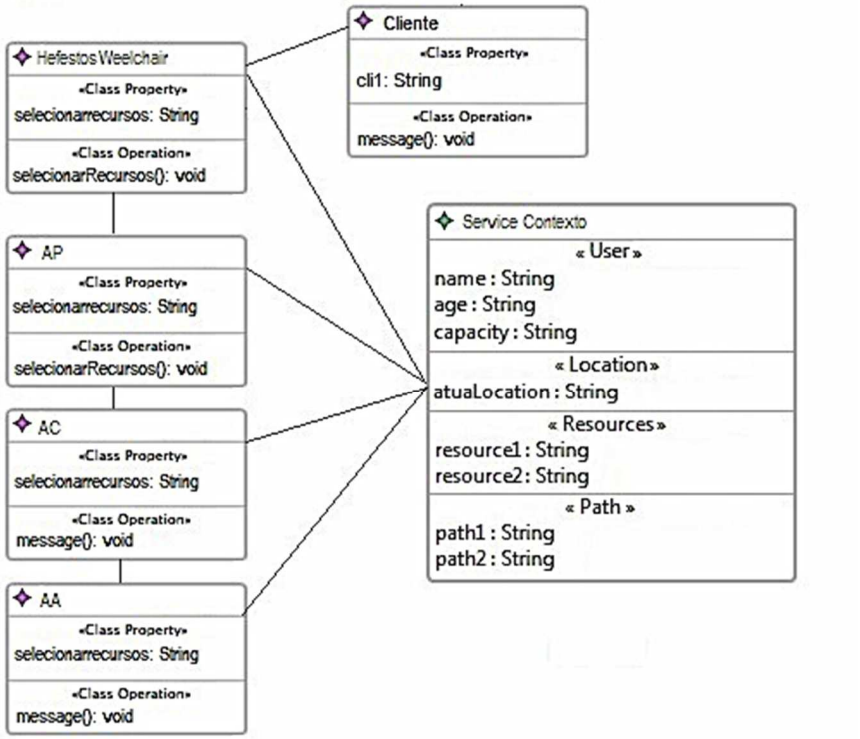
Fonte: Tavares (2012).

Figura 38: Modelagem do Hefestos em UML pura



Fonte: Elaborado pela autora.

Figura 39: Modelagem do Hefestos em UML2Context



Fonte: Elaborado pela autora.

6.3.5 Aplicação SAUM

O SAUM, *Sistema de Assistência a Urgência Médic*a, (NOVA JR., 2008) é uma dissertação de mestrado, defendida em 2008 na Universidade Federal de Pernambuco, na área de Saúde Ubíqua. O trabalho propõe um sistema ubíquo para ambiente hospitalar de urgência que visa deixar mais simples e prático a comunicação entre as equipes paramédicas móveis e os médicos de plantão em emergências hospitalares (NOVA JR., 2008).

Os paramédicos em UTIs móveis, médicos de plantão e computadores de diversos hospitais estão interligados a uma central onde poderão compartilhar diversas informações do paciente para facilitar a identificação do paciente e sua situação de saúde e pelo sistema localiza-se um hospital com vaga para receber o paciente. Se o paramédico já esgotou suas ações do que pode ser feito no atendimento, de posse do seu *Palmtop,* o paramédico entra em contato, através do *SAUM,* diretamente com o *Palmtop* do médico de plantão no hospital destino. De pronto o médico recebendo a solicitação, levanta por meio do sistema todos os procedimentos realizados pelo paramédico, de posse de todas estas informações o médico sugere outro procedimento, agora via voz, e solicita a imagem do enfermo, enquanto o paramédico ouvindo as instruções do médico de plantão realiza os procedimentos, o enfermeiro assistente, que está na UTI móvel, liga a câmera do *Palmtop* e em tempo real filma todo o procedimento que está sendo realizado pelo paramédico, sob as instruções do médico de plantão.

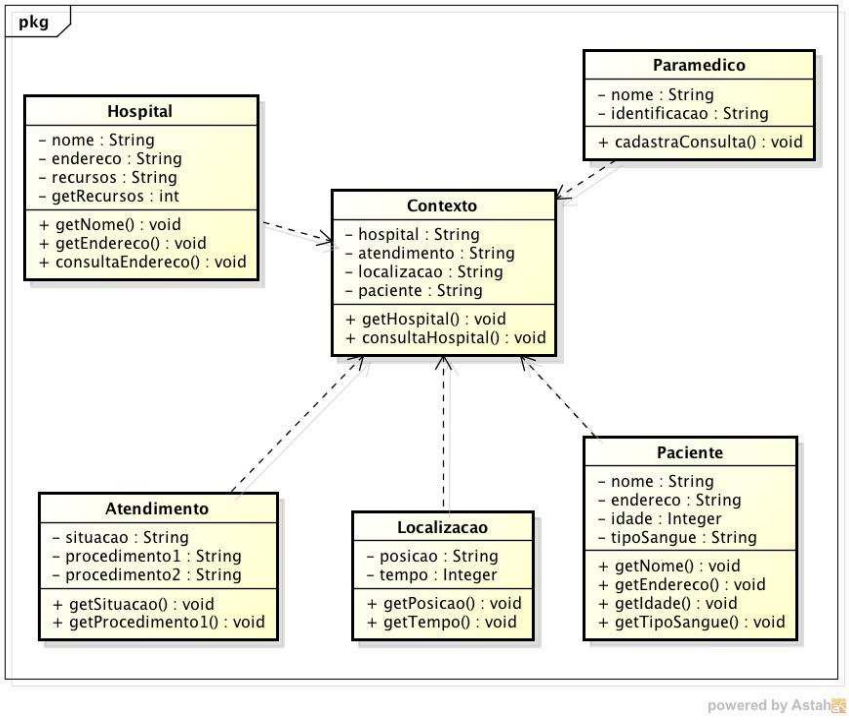
O próprio médico que estava passando as instruções já informa à ambulância que o determinado hospital onde se encaminharia o enfermo não tem mais vagas. Desta forma, o próprio médico, através do *palmtop* e utilizando o *SAUM* convoca outro médico de plantão

em outro hospital, mas agora não para o *palmtop* do outro médico e sim para o computador central do outro hospital. Este já sabendo disponibilidade da vaga, informa a aceitação do novo paciente, e por meio de mensagem no telão central da emergência informa a chegada em cinco minutos desta ambulância informando as condições atuais de saúde deste paciente como também disponibilizando todo prontuário médico e procedimentos realizados pela equipe de paramédicos que o encaminha.

Figura 40: Ilustração geral de funcionamento do SAUM

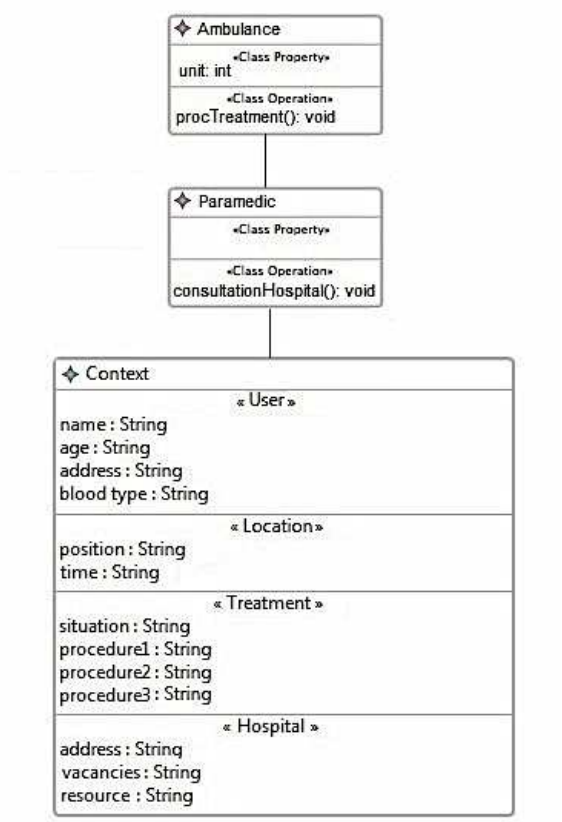
Fonte: Nova Jr. (2008).

Figura 41: Modelagem do SAUM em Astah



Fonte: Elaborado pela autora.

Figura 42: Modelagem do SAUM em UML2Context



Fonte: Elaborado pela autora.

Nesta situação de atendimento do SAUM existe a utilização de pelo menos três cenários que podem ser observados na Figura 41. O primeiro é a comunicação da UTI móvel e o médico, onde o paramédico solicita melhores instruções para proceder no atendimento. O segundo cenário é o acompanhamento médico remoto, onde o médico passa as instruções por voz e solicita a filmagem em tempo real transmitida para o *Palmtop* do médico. O terceiro cenário é a comunicação entre o *Palmtop* e o computador central do hospital. A Figura 42 esboça o contexto extraído dos elementos mencionados no trabalho e modelado no Astah. A Figura 43, por sua vez, apresenta o mesmo contexto modelado no UML2Context.

6.4 Resultados Obtidos

Com o resultado das questões que foram submetidas a profissionais de mercado e estudantes, buscou-se investigar o impacto em relação aos esforços para interpretação e identificação dos modelos para efetuar uma implementação de contexto. Além disso, o resultado também mostra a eficácia da linguagem de modelagem UML2Context como facilitadora na implementação dos modelos de contexto. Para tanto os resultados foram cadastrados e submetidos a uma ferramenta de estatística descritiva RStudio[7], para geração dos resultados e tabelas para a análise. No geral, a avaliação conclui que o modelo UML2Context alivia o esforço para detectar a melhor implementação e simplifica o entendimento sobre os modelos de contexto.

[7] RStudio: http://erwin.com/products/data-modeler

As avaliações tiveram os resultados resumidos nas Tabela 4 e 5. Nas mesmas os resultados estão separados pela variável independente linguagem de modelagem, que separa UML2Context e UML pura, mostrada na coluna “Tratamento”.

Tabela 4: Tabela de variáveis qualitativas resultados da avaliação

Questões	Variáveis	Tratamento	Média	% diff
Todas	Taxa de respostas corretas	UML	0,53	28,41%
		UML2Context	0,73	
	Esforço de interpretação	UML	6,06	61,03%
		UML2Context	2,36	
	Taxa de má interpretação	UML	0,50	35,98%
		UML2Context	0,79	
Q1	Taxa de respostas corretas	UML	0,71	5,88%
		UML2Context	0,67	
	Esforço de interpretação	UML	8,00	75,00%
		UML2Context	2,00	
	Taxa de má interpretação	UML	0,71	2,86%
		UML2Context	0,73	
Q2	Taxa de respostas corretas	UML	0,25	70,00%
		UML2Context	0,83	
	Esforço de interpretação	UML	5,64	60,99%
		UML2Context	2,20	
	Taxa de má interpretação	UML	0,10	88,37%
		UML2Context	0,90	
Q3	Taxa de respostas corretas	UML	0,67	11,11%
		UML2Context	0,75	
	Esforço de interpretação	UML	4,48	50,91%
		UML2Context	2,20	
	Taxa de má interpretação	UML	0,77	2,63%
		UML2Context	0,79	
Q4	Taxa de respostas corretas	UML	0,54	18,75%
		UML2Context	0,67	
	Esforço de interpretação	UML	6,20	61,29%
		UML2Context	2,40	
	Taxa de má interpretação	UML	0,50	31,43%
		UML2Context	0,73	
Q5	Taxa de respostas corretas	UML	0,46	38,89%
		UML2Context	0,75	
	Esforço de interpretação	UML	5,96	49,66%
		UML2Context	3,00	
	Taxa de má interpretação	UML	0,44	44,74%
		UML2Context	0,79	

Fonte: Elaborado pela autora.

A Tabela 4, além de separar as informações por questões aplicadas na avaliação, mostra para cada questão as variáveis de avaliação: “Taxa de respostas corretas”, “Esforço de interpretação” e “Taxa de má interpretação”; Apresentando na coluna “Média” os resultados individuais das variáveis. Pode-se ver na coluna “diff” a diferença percentual dos valores para cada tratamento. E no início da tabela uma compilação de todas as questões na opção “Todas”. Com os resultados coletado pode-se observar que houve uma diferença percentual significativa em relação aos dois tratamentos em cada questão da avaliação. Os resultados compilados de todas as questões indicam que a UML2Context aumentou a taxa de respostas corretas em 28.41%, reduziu o esforço de interpretação em 61.03% e melhorou a interpretação dos modelos em 35.98% se mostrando realmente eficiente.

Na Tabela 5 pode-se observar as variáveis de quantificação apresentando os resultados por modelo, a quantidade de participantes total, a Taxa de Respostas Corretas (TRC), a Taxa de Má Interpretação (TMI) e o Esforço de Interpretação (EI) para cada questão de avaliação. Olhando para os resultados da tabela, pode-se ver na coluna TRC que a *Taxa Respostas Corretas* na modelagem feita pela UML2Context foi em geral maior do que a quantidade de acertos da modelagem em UML pura, com uma única exceção. E também que o tempo em minutos investido para identificação da implementação correta para os modelos de contexto diminuiu na UML2Context. A coluna TRC mostra que a *Taxa de Má Interpretação* é maior na UML2Context, indicando que há menos divergência nas respostas deste modelo. E por fim na coluna EI que mostra o *Esforço de Interpretação*, em minutos, vemos que o esforço diminui significativamente nos modelos UML2Context.

Tabela 5: Relação das variáveis de quantificação

		Participantes	TRC	TMI	EI (min.)
UML	Questão 01	24	0,71	0,71	8,00
	Questão 02	24	0,25	0,10	5,64
	Questão 03	24	0,67	0,77	4,48
	Questão 04	24	0,54	0,50	6,20
	Questão 05	24	0,46	0,44	5,96
UML2Context	Questão 01	24	0,67	0,73	2,00
	Questão 02	24	0,83	0,90	2,20
	Questão 03	24	0,75	0,79	2,20
	Questão 04	24	0,67	0,73	2,40
	Questão 05	24	0,75	0,79	3,00

Fonte: Elaborado pela autora.

Com os dados coletados e inseridos na ferramenta RStudio de estatística descritiva foi feito os cálculos para cada variável dividido por tratamento, gerando então os resultados da Tabela 6. Nesta tabela constam os resultados para as variáveis de quantificação, “TRC” (Total de Respostas Corretas), que mostra o índice global de identificação da melhor implementação para a modelagem do contexto, quanto maior o índice maior é a capacidade de identificação da resposta correta. Mostra também a variável “EI” (Esforço de Interpretação) que mostra a média de esforço de interpretação com o tempo calculado em minutos que os indivíduos utilizaram para analisar os modelos e identificar as questões em cada modelagem. E a variável “TMI” (Taxa de Má Interpretação), que avalia o quanto o modelo desviou do raciocício correto. Foram calculados o desvio padrão, o mínimo, máximo, a média e a mediana. Por fim, nas duas últimas colunas da tabela mostram os cálculos:

- O Student`s t-test, que é um teste de dados emparelhados, muito usado quando precisamos comparar médias de amostras diferentes. Um t-teste é um teste de hipótese estatístico em que a estatística de teste segue a distribuição t de Student para avaliar se a hipótese nula é suportada. Ele pode ser usado para determinar se dois conjuntos de dados são significativamente diferentes uns dos outros, e é o mais frequentemente aplicado quando a estatística de teste que seguem uma distribuição normal ou quando o termo de escala é desconhecida e é substituído por uma estimativa com base nos dados.
- Teste de Wilconxon, que é um teste de hipótese estatística não-paramétrica usada quando se comparam duas amostras relacionadas, amostras pareadas ou repetidas medições em uma única amostra para avaliar se a sua média populacional difere, ou seja, é um teste de diferença emparelhada muito utilizado como alternativa para o t-teste de pares combinados.

Tabela 6: Estatística descritiva e testes estatísticos para medidas

Variáveis	Tratamento	Desvio padrão	Mínimo	Mediana	Máximo	Média	% diff	Wilcoxon	Paired t-teste
								p-value	p-value
Taxa de respostas corretas	UML	0,18	0,25	0,54	0,7	0,53	28,41 %	0,03	0,06
	UML2 Context	0,07	0,66	0,75	0,77	0,73			
Esforço de interpretação	UML	1,27	2	5,9	8	6,06	61,03 %	0,02	0,002
	UML2 Context	0,38	4,4	2,2	3	2,36			
Taxa de má interpretação	UML	0,26	0,1	0,5	0,77	0,50	35,98 %	0,02	0,05
	UML2 Context	0,07	0,72	0,79	0,89	0,79			

Fonte: Elaborado pela autora.

Os testes de normalidade de Shapiro-Wilk e Kolmogorov-Smirnov (disponível em Levine et al. (1999)) indicaram que os dados estão normalmente distribuídos. Logo, o t-test emparelhado é aplicado para testar as hipóteses. Além disso, o Wilconxon teste foi aplicado para reduzir a ameaça a validade estatística dos testes das hipóteses. Esses testes serão utilizamos nas próximas subseções para o teste das hipóteses levantadas. Assim, de posse dos dados coletados e distribuídos nas três tabelas para potencializar a sua visualização, será feito agora a análise de cada questão e suas hipóteses levantada.

6.4.1 RQ1: QP1: Taxa de Respostas Corretas

Na primeira questão de pesquisa, investiga-se o impacto da UML2Context na taxa de acertos por questões respondidas. Visualizando os dados da Tabela 6 pode-se ver as médias onde a taxa de acertos ficou superior nos modelos UML2Context. Desenvolvedores detectam, em média, em cerca de 28,41 por cento mais acertos nos modelos UML2Context do que os modelos com UML Pura, isto é, uma média de 0,53 (UML), em comparação com uma média de 0,73 (UML2Context), o que também se confirma observando as medianas. Na Figura 44 é possível ver que a taxa de respostas corretas na UML2Context é bem superior se comparado com a UML Pura, totalizando as 5 questões da avaliação e que trata a Tabela 5.

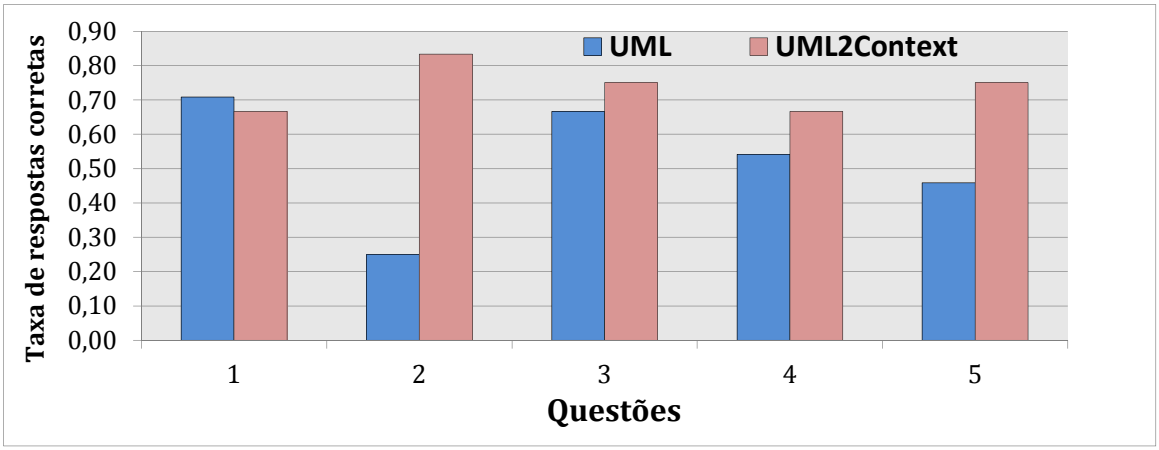
Analisando a $H_{1\text{-}0}$, tem-se para investigar a hipóteste nula, o t-teste emparelhado e o teste não paramétrico de Wilcoxon que pode ser visto na Tabela 6. A estatísticas t coletada é

0,06 o que indica que a hipótese nula pode ser rejeitada. Implica que a diferença média da taxa de respostas corretas em modelos UML2Context e UML pura não é zero. Portanto, vê-se que os desenvolvedores identificam a implementação correta de contexto com mais facilidade nos modelos UML2Context do que em modelos em UML pura. A média das diferenças comentada anteriormente entre os pares de modelos indicam a direção em que o resultado é significativo. Também na Tabela 6 é possível analisar o teste não paramétrico de Wilcoxon que é aplicada para eliminar qualquer ameaça relacionada com a conclusão validade estatística, onde podemos ver que o baixo valor do valor-p = 0,03 coletado também confirma que podemos rejeitar a hipótese nula H1-0.

Analisando a H1-1 vê-se que a hipótese foi rejeitada, pois a taxa de acertos da UML2Context (0,73) não é menor do que a taxa da UML pura (0,53). Conclui-se que a UML2Context não dificulta a identificação dos modelos.

Colocando em escala os resultados da análise de taxa de respostas corretas da Tabela 5 na Figura 44 pode-se visualizar de forma mais nítida a diferença da taxa de respostas corretas da UML2Context contra a UML Pura para cada uma das 5 questões, confirmando os resultados acima.

Figura 43: Taxa de respostas corretas



Fonte: Elaborado pela autora.

Então respondendo a primeira questão podemos dizer que a UML2Context afeta a eficiência de desenvolvedores para identificar a melhor forma de implementação do contexto positivamente aumentando a taxa de respostas corretas.

6.4.2 QP2: Esforço de Interpretação

Na segunda questão de pesquisa, investiga-se o impacto da UML2Context no esforço dos desenvolvedores para identificar a implementação correta. Visualizando os dados das Tabelas 4 e 5 pode-se ver que os indivíduos gastam mais esforço para detectar as respostas corretas nos modelos UML pura do que nos modelos UML2Context. A Tabela 5 mostra o esforço médio por questão e na Tabela 4 pode-se ver no primeiro registro em “Todas” na linha de esforço de interpretação, a média de esforço empregado para identificação de uma resposta é de 6,06 (minutos) em modelos de UML Pura e 2,36 em modelos UML2Context. Este esforço menor no uso da UML2Context também é observado na Tabela 6 onde mostra a

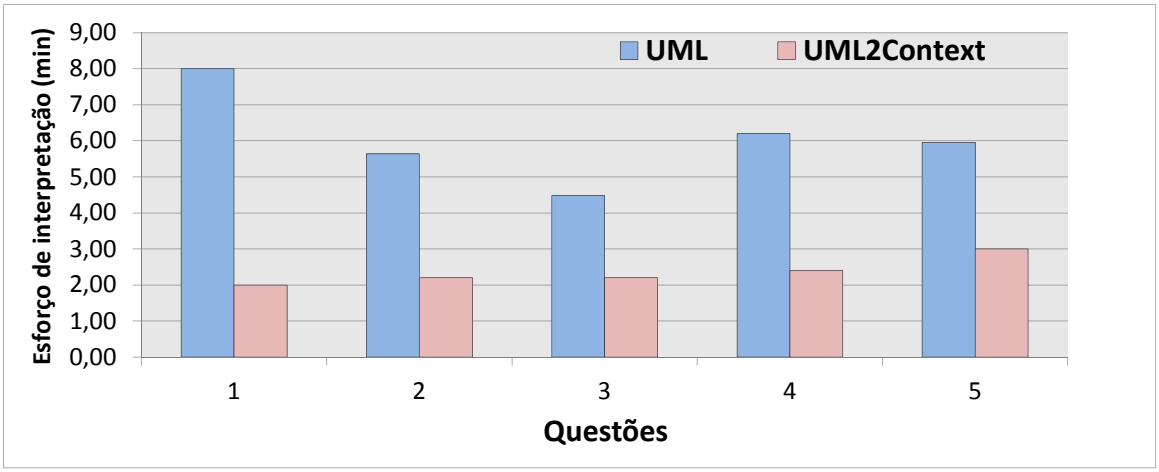
diferença de 61,03% no esforço, ou seja, o esforço empregado para se chegar a uma resposta na UML2Context é 61,03% menor do que usando a UML Pura. Este esforço menor no uso da UML2Context também é observado comparando as medianas de 5,9 para a UML pura e apenas 2,2 para a UML2Context e que se repetem nas médias.

Analisando a H2-0 tem-se para a hipóteste nula o t-teste emparelhado que pode ser visto na Tabela 6. É possível verificar que a estatística t coletada é 0,002. Este valor pequeno indica que a segunda hipótese nula (H2-0) pode ser rejeitada. Isto sugere que a diferença média de esforço para chegar a uma solução em modelos UML2Context e UML Pura não é zero. Assim, há evidências de que os desenvolvedores investem mais esforço para detectar inconsistências nos modelos UML Pura do que em modelos UML2Context. Também o teste não paramétrico de Wilcoxon é aplicada para eliminar qualquer ameaça relacionada com a conclusão da validade estatística. O baixo valor do valor-p coletado de 0,02 também confirma a conclusão de que podemos rejeitar a hipótese nula H2-0. Vemos que essa hipótese é rejeitada uma vez que o esforço para identificar a implementação correta na UML2Context não é igual ou maior do que em UML pura. Isso indica que os desenvolvedores não fazem um esforço maior para analisar os modelos na UML2Context.

Para analisar a H2-1 são considerados os dados da Tabela 4. Pode-se concluir que o tempo médio para chegar a uma resposta é bem menor na UML2Context, ficando em 2,36 minutos, enquanto que na UML pura o tempo médio para a escolha de uma resposta ficou em 6,06 minutos. Essa taxa de esforço é mostrada também na Tabela 5 onde pode-se verificar que o esforço na UML pura por questão é maior que na UML2Context. Com isto é possível identificar que os desenvolvedores de posse dos modelos UML2Context conseguiram chegar a uma conclusão mais rapidamente. Verifica-se nessa hipótese que o esforço para identificar uma resposta para a implementação na UML2Context é menor 61,03% do que na UML pura. Assim pode-se concluir que essa hipótese é valida e os desenvolvedores fizeram menos esforço com os modelos UML2Context.

Colocando em escala os resultados da análise de taxa de esforço empregado da Tabela 5, obtem-se o gráfico da Figura 45. Na figura observa-se de forma mais nítida a diferença de esforço para compreensão e análise para chegar a uma resposta, entre a UML2Context e a UML Pura. Pode-se notar que no caso da UML Pura o esforço ficou bem superior confirmando as discussões anteriores.

Figura 44: Taxa de esforço empregado



Fonte: Elaborado pela autora.

Respondendo então a segunda questão a UML2Context influencia nos esforços investidos pelos desenvolvedores para implementar corretamente o modelo de contexto diminuindo o esforço empregado.

6.4.3 QP3: Taxa de Má Interpretação

Investigou-se o impacto da UML2Contex na taxa de má interpretação dos desenvolvedores nos modelos de contexto. Para chegar a esses resultados, como descrito anteriormente, foi utilizada a taxa “TMI”; onde TMI = 0 se as respostas são distribuídos aleatoriamente sobre todas as opções sem padrão ao longo das cinco alternativas, então os modelos causam interpretações erradas; se TMI = 1 as respostas estão concentradas em apenas uma opção, nesse caso concluímos que as más interpretações dos diagramas não levam a erros de interpretação. Na Tabela 5 pode-se ver que os indivíduos tiveram um TMI mais elevado nas questões que utilizaram a UML2Context. E na Tabela 4 observa-se que o TMI teve um aumento de 35,98% na UML2Context, onde a média ficou em 0,79 na UML2Context contra 0,50 na UML Pura. Estes valores também podem ser vistos na mediana na Tabela 6. A média superior quando os indivíduos se concentram em uma opção de resposta mostra que os modelos não são ambíguos e mantém uma linha de interpretação. Sendo assim conclui-se que na UML2Context o raciocínio dos desenvolvedores fica mais alinhados e as suas interpretações são mais acertadas chegando as mesmas conclusões em comparação com os modelos na UML pura.

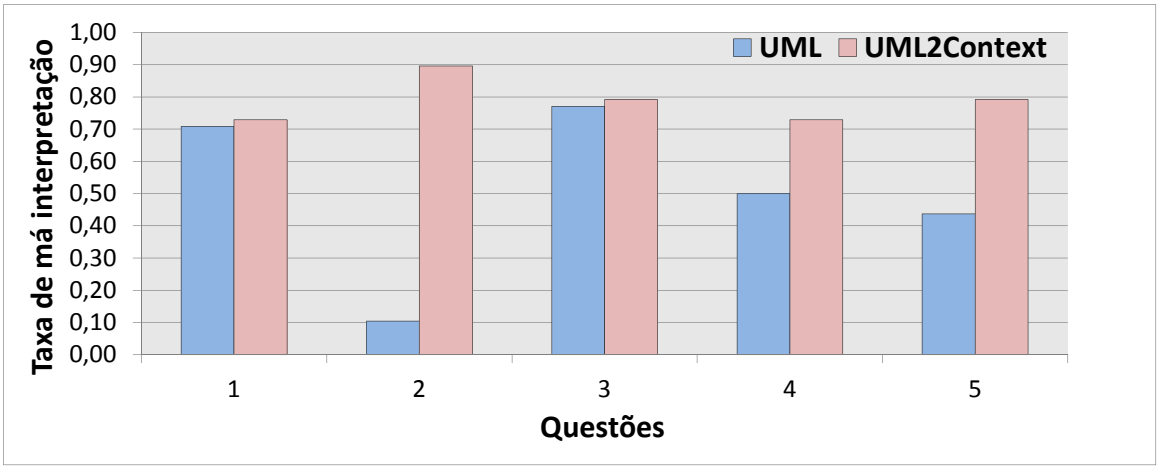
Analisando a H3-0 também com o t-teste emparelhado pode-se ver na Tabela 6 que a taxa de erros de má interpretação na UML2Context é maior em 35% do que na UML pura. As médias de TMI de 0,79 na UML2 Context e 0,50 na UML Pura indicam que no primeiro modelo existe uma linha de pensamento que leva as mesmas conclusões, enquanto que a taxa menor na UML Pura indica que as respostas dos desenvolvedores se distribuíram-se mais aleatóriamente, sem padrão, ao longo das cinco alternativas, tendo mais probabildiade dos modelos UML puros causarem interpretações erradas. Também aplicando o teste não paramétrico de Wilcoxon para verificar essa conclusão, vemos o valor de p é 0,02. Assim, como o valor p é menor que 0,05, pode-se concluir que há evidências de que o TMI em modelos UML2Context é significativamente menor do que nos modelos UML Pura, portanto rejeitamos a hipótese nula H3-0.

Analisando a H3-1 pode-se ver que esta hipóteste é nula uma vez que a taxa na UML2Context não é menor que na UML pura. A taxa de 0,79 na UML2Context contra a taxa de 0,50 na UML Pura indica que nos modelos UML2Context os desenvolvedores foram induzidos a chegarem na mesma linha de raciocínio, levando a menos erros de interpretação.

Colocando em escala os resultados da análise de taxa de má interpretação da Tabela 5 na Figura 46 pode-se visualizar de forma mais nítida a diferença da taxa entre a UML2Context e a UML Pura, distribuída nas 5 questões avaliadas. Quanto maior a taxa, melhor é o índice interpretação dos modelos. Isto então confirma a discussão anterior de que a taxa superior do UML2Conext indica menos erros de interpretação nos modelos considerados.

Então respondendo a terceira questão a UML2Context reduz evidentemente a má interpretação na modelagem de contexto em comparação com a modelagem em UML pura.

Figura 45: Taxa de má interpretação



Fonte: Elaborado pela autora

6.4.4 Discussão

Nessa avaliação buscou-se em cada questão explorar diferentes situações para que se obtivesse um panorama bem diversificado para responder as questões levantadas. No entanto, fazendo um apanhado geral nesta seção é possível concluir que de acordo com os resultados de todas as questões impressionantemente a UML2Context aumentou a taxa de respostas corretas em 28,41%, reduziu o esforço de interpretação em 61,03% e melhorou a interpretação dos modelos em 35,98%. Os resultados sugerem que a modularização das informações de contexto em um novo conceito chamado de UML2Context traz benefícios, quando comparada com a decomposição de tais informações com a UML Pura.

Analisando as respostas é percebido que pequena parcela dos indivíduos se dispuseram a fazer comentários sobre as questões, os poucos que o fizeram foi para justificar alguma resposta. Como já discutido anteriormente os indivíduos poderiam sentir-se pressionados a fazer uma implementação mesmo com dúvidas ou erros, no entanto houve apenas dois comentários sobre as questões nesse sentido, onde os indivíduos apontaram uma reposta esclarecendo sua decisão. Assim esta possibilidade não é uma ameaça ou altera o resultado da avaliação. De qualquer forma a TMI de 35,98% é uma diferença significativa indicando que a taxa de má interpretação teve esse aumento para a UML Pura.

A taxa de respostas corretas (TRC) foi de 28,41% maior na UML2Context, deixando evidente que sua simplicidade e objetividade no tratamento do contexto ajudou os indivíduos em manter uma linha de entendimento e localizar uma resposta. No entanto uma taxa bem expressiva foi a de esforço (EI) chegando a uma diferença de 61,03% menor de esforço nos modelos UML2Context, reduzindo assim drasticamente o tempo que os indivíduos necessitaram para entendimento do modelo de contexto.

Na avaliação foi abordado dois grupos diferentes, primeiramente profissionais de TI e posteriormente estudantes de graduação ou mestrado. Observamos nestes dois grupos que, nos grupos dos profissionais que atuam no mercado a tendência foi maior de escolher uma implementação e em alguns casos justificar a resposta mesmo tendo algum item que eles não concordavam. No grupo de estudantes a tendência maior foi de escolher a opção que nenhuma resposta é correta por haver problemas nos modelos, também com algumas justificativas. No entanto o grupo de profissionais, em geral, levou um tempo maior para responder as questões do que o grupo de estudantes. Pode-se avaliar que os profissionais levam mais tempo

avaliando as opções e ocorre um sentimento de que devem escolher uma opção enquanto que os estudantes ou não se sentem pressionados a escolher uma opção ou não avaliam o tempo suficiente para decidirem por uma questão. No entanto também foi observado que dentro do grupo de estudantes houve muito mais retorno de observações escritas.

Em relação aos dois questionários foi observado também que houve mais observações escritas para o questionário com o modelo UML pura. O que faz sentido de acordo com nossos resultados onde vimos que a taxa de má interpretação indicou que na UML pura os indivíduos dispersam mais suas respostas, indicando que o modelo não remente a mesma linha de raciocínio.

Quanto a validade da conclusão estatística, pode-se afirmar que diretrizes experimentais foram seguidos para eliminar esta ameaça onde os pressupostos dos testes estatísticos (teste t pareado e Wilcoxon) não foram violados. Conjuntos de dados coletados foram distribuídos normalmente. A homogeneidade dos indivíduos foi assegurado. O método de quantificação foi devidamente aplicado com ferramenta estatística. Os testes de Kolmogorov-Smirnov e Shapiro-Wilk foram utilizados pela ferramenta estatística para confirmar a probabilidade de que a amostra recolhida foi distribuída normalmente.

Finalizando pode-se dizer que as experiências acumuladas demonstram que, com a execução da avaliação e seus resultados vemos que a UML2Context afeta positivamente na análise para a correta escolha de implementação pelos indivíduos, e assim levando a melhor interpretação de contexto. Sendo assim pressupõe-se que, a contínua expansão da modelagem causa impacto direto na avaliação dos conhecimentos estratégicos sobre contexto, para que assim os desenvolvedores possam atingir a excelência no desenvolvimento de sistemas cientes de contexto.

7 CONCLUSÃO

Pode-se afirmar que a falta de uma modelagem específica e um ambiente que dê suporte a linguagem de modelagem contexto, que possa ser usada no desenvolvimento de aplicações cientes de contexto, representa um ponto chave na decisão do uso eficiente ou não da modelagem e da ferramenta que é proposta. Cabe destacar também que modelar contexto não é uma tarefa simples e nem todas as modelagens de contexto contém os detalhes necessários para utilização ampla no desenvolvimento das aplicações cientes de contexto.

Essa dissertação apresentou a definição de uma linguagem para modelagem de contexto denominada UML2Context, para auxiliar na modelagem de sistemas cientes de contexto. Além disso, como forma de avaliar o modelo, foi desenvolvida uma ferramenta, a UML2Context Tool, que fornece uma perspectiva diferente de trabalho com o contexto. Embora a proposta, através da linguagem de modelagem, já proporcione uma nova visão e entendimento do uso do contexto nas aplicações cientes de contexto, sem o apoio de uma ferramenta, tornaria o trabalho mais difícil de ser realizado. Já que as aplicações padrões que utilizam UML não suportam todos os conceitos levantados e a representação apenas conceitual, sem o uso de gráficos, não seja tão eficiente para os desenvolvedores. Os benefícios obtidos com a linguagem *UML2Context* e a ferramenta *UML2Context Tool* emergem diante da resolução de alguns desafios encontrados até então em ambiente acadêmico e profissional e que seguem descritos.

Cabe destacar a importância de conhecer a modelar contextos em detalhes, pois implica em conhecer todos os termos e conceitos relacionados ao mesmo. Não apenas os conceitos e os relacionamentos definidos no metamodelo proposto como também às restrições aplicadas aos elementos. Caso contrário, um modelo criado manualmente pode apresentar inconsistência ao ponto de torná-lo inválido. Como a ferramenta engloba muitos dos conceitos tipicamente envolvidos como contextos, e faz a exibição de diagramas, permite que desenvolvedores não experientes modelem diagramas consistentes. Principalmente em ambiente profissional onde os desenvolvedores não dispõem do tempo necessário para a ambientação aos conceitos do contexto para desenvolvimento de um software ciente de contexto.

Com os resultados da avaliação ficou evidente que para as aplicações consideradas a UML2Context aumentou a taxa de respostas corretas identificadas em 28,41%, contribuindo para uma melhor implementação. Também reduziu o esforço de interpretação em 61,03%, diminuindo significativamente o tempo que os indivíduos levaram para entender o modelo de contexto e localizar uma implementação, e ainda assim chegaram na implementação mais adequada. Além disso, a UML2Cotnext melhorou a interpretação dos modelos em 35,98%, fazendo com que os indivíduos não oscilassem muito em suas respostas mantendo a linha de pensamento. Como a taxa de respostas corretas foi alta conclui-se que os desenvolvedores mantiveram a linha correta de raciocínio. Portanto, os resultados sugerem que a modularização das informações de contexto em um novo conceito chamado de UML2Context traz benefícios significativos, comparada com a modelagem feita na UML Pura.

A principal contribuição científica se deu pela proposição de uma linguagem de modelagem de contexto, baseada em ontologias e que abrange requisitos para a aplicações cientes de contexto. O trabalho contribui efetivamente nos padrões de representação ou representações pré-definidas para o modelo de contexto. Assim evitando problemas onde o padrão não é seguido, evitando comprometer a compreensão e o bom entendimento dos modelos, além de reduzir ambiguidade e inconsistências tão comuns em projetos [Farias et al., 2009]. A linguagem UML2ContextTool é uma ferramenta para uso dos desenvolvedores

onde os modelos possuem uma representação gráfica padrão dos elementos que são utilizados no desenvolvimento da aplicação, tornando o trabalho de desenvolvimento eficiente e eficaz e os modelos de fácil compreensão. O destaque frente aos trabalhos relacionados ocorre tendo em vista que reúne pontos positivos de cada um deles, focando na modelagem de contexto de forma a trazer a mesma para a realidade do desenvolvimento. A UML2Context estende a UML como no Camel, UML CaProf e a MAS-ML; A UML2Context utiliza assim como o Camel a linguagem de metamodelo ECORE um padrão confiável; Utiliza a modelagem EMF como o Método de Divisão; A ferramenta Eclipse assim como o Camel e cria uma linguagem de modelagem assim como na MAS-ML; Trabalha com diagramas de classe assim como a maioria dos trabalhos e utiliza a abordagem diferenciada da UML estendida.

Trabalhou-se na hipótese de estabelecer uma linguagem de modelagem estendendo a UML e como resultado, pode-se observar que a linguagem UML2Context simplifica a modelagem de contexto já estabelecendo um padrão para modelagem. Os elementos principais estão destacados, e com seus atributos definidos, facilitando o entendimento dos desenvolvedores, e também permite a extensão de novos elementos.

Esse trabalho apresenta algumas oportunidades de trabalhos futuros em termos de evoluções na ferramenta e de modelagem de outros diagramas. Baseado nas limitações atuais da ferramenta, a versão atual apenas contempla o diagrama de classes. Portanto, uma possibilidade é estender a ferramenta para incorporar os demais modelos estáticos da linguagem, possibilitando várias visões do contexto e seus elementos. É possível também que a ferramenta seja adaptada para efetuar a geração de código.

Outra possibilidade de trabalho futuro é a incorporação na ferramenta de um controle das inconsistências em relação ao apresentado no *Modeling View* que pode ser mostrada em uma aba chamada *Problems View*. Esta funcionalidade pode viabilizar o uso de modelagem de contexto dentro do desenvolvimento dirigido por modelos, na qual modelos são vistos como artefatos de primeira ordem. O *Problems View* teria o objetivo de disponibilizar uma funcionalidade de validar o modelo em relação ao metamodelo da linguagem. Se existir alguma inconsistência, ela será mostrada nesse componente.

Por fim, pretende-se no futuro utilizar o modelo proposto na modelagem de protótipos ubíquos desenvolvidos no PIPCA da Unisinos. Com isso, avaliar a aplicabilidade do modelo em projetos científicos em desenvolvimento analisando o impacto do UML2Context na simplificação do processo de modelagem de sistemas cientes de contexto.

REFERÊNCIAS

ABOWD, Gregory D., HAYES, Gillian R., IACHELLO, Giovanni, KIENTZ, Julie A., PATEL, Shwetak N., STEVENS, Molly M., TRUONG, Khai N.. Prototypes and Paratypes: Designing Mobile and Ubiquitous Computing Applications. **Magazine IEEE Pervasive Computing**, vol. 4, no. 4, p 67-73, dec. 2005.

AHLUWALIA, Punit, VARSHNEY, Upkar, KOONG, Kai S., WEI, June. Ubiquitous, mobile, pervasive and wireless information systems: current research and future directions. **Journal International of Mobile Communications**, vol. 12, no. 2, p. 103-141, mar. 2014.

ASTAH. Ferramenta de Modelagem. Disponível em: <http://astah.net/>. Acesso em: 16 dez. 2015.

BAUER, Joseph. *Identification and Modeling of Contexts for Different Information Scenarios in Air Traffic.* 2003. Diplomarbeit, Technische Universita¨t Berlin Fakulta¨t IV - Elektrotechnik und Informatik Institutfu¨rComputergestu¨tzteInformationssysteme, Berlin.

BENSELIM, Mohamed Salah, SERIDI-BOUCHELAGHEM, Hassina. Extending UML Class Diagram Notation for the Development of Context-aware Applications. **Journal of Emerging Technologies in Web Intelligence**, vol. 5, no. 1, p. 35-44, fev. 2013.

BOOCH, Grady RUMBAUGH, James, JACOBSON, Ivar. UML – Guia do Usuário (O mais avançado tutorial sobre Unified Modeling Language (UML) elaborado pelos próprios criadores da linguagem). Rio de Janeiro: Campus, 2000. p. 12-550.

BULCÃO Neto, Renato de Freitas. *Um processo de software e um modelo ontológico para apoio ao desenvolvimento de aplicações sensíveis ao contexto*. 2006. Tese, apresentada ao Instituto de Ciências Matemáticas e de Computação da USP, São Paulo.

BULCÃO Neto, R. F., PIMENTEL, M. G. C.. Toward a Domain-Independent Semantic Model for Context-Aware Computing. In: Proceedings of the Third Latin American Web Congress (LA-WEB'05), 2005, Buenos Aires, Argentina. **Anais eletrônicos**… IEEE, 2005. Disponível em: <http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=10609>. Acesso em: 15 jan. 2014.

CARTON, A., CLARKE, S., SENART, A., CAHILL, V.. Aspect-Oriented Model-Driven Development for Mobile Context-Aware Computing. In: First International Workshop on Software Engineering for Pervasive Computing - Applications, Systems, and Environments (SEPCASE'07), 2007. **Anais eletrônicos**… New York: ACM DL, 2007. Disponível em: <http://dl.acm.org/citation.cfm?id=1270346>. Acesso em 20 jan. 2014.

COSTA, C. A., BARBOSA, J. and LOPES, J., SOUZA, R., GEYER, C., GUSMÃO, M., YAMIM, A.. A Model for Context Awareness in Ubicomp. In: WebMedia`12 - Proceedings of the 18th Brazilian symposium on Multimedia and the web, 2012, N. **Anais eletrônicos**… New York: ACM DL, 2012. Disponível em: <http://dl.acm.org/citation.cfm?id=2382672>. Acesso em: 20 jan. 2014.

COSTA, C. A., SILVA, L. C., BARBOSA, J. L. V., YAMIM, A. C., GEYER, C. F. R.. A primer of ubiquitous computing challenges and trends. In: F. M. M. Neto and P. F. R. Neto, editors, Designing Solutions-Based Ubiquitous and Pervasive Computing: New Issues and Trends, vol. 1, chapter 15, IGI Global Publishing, Hershey, 2010. P. 282–30.

COSTA, C. A., YAMIM, A. C., GEYER, C. F. R.. Toward a General Software Infrastructure for Ubiquitous Computing. **IEE Pervasive Computing,** vol. 7 no.1, p. 64-73, mar. 2008.

COSTA, H. J. M.. **Um Modelo de Arquitetura para o Turismo Ubíquo Utilizando Dispositivos Móveis**. 2013. 81 f. Dissertação de Mestrado do programa de pós-graduação em Computação Aplicada (PIPCA) da Universidade do Vale do Rio dos Sinos, São Leopoldo, RS, 2013.

DEY, Anind K.. Understanding and Using Context. College of Computing e GVU Center, Georgia Institute of Technology, Atlanta, GA, USA. **Journal Personal and Ubiquitous Computing**, vol. 5, issue 1, p.4-7, fev. 2001.

DEY, A. K. ABOWD, G. D. The Context Toolkit: Aiding the Development of Context-Aware Applications. In: Proceedings of the 3rd International Symposium on Wearable Computers, 2000. **Anais eletrônicos…** Atlanta: Georgia Institute of Technology, 2000. Disponível em: <http://www.cc.gatech.edu/fce/contexttoolkit/pubs/chi99.pdf>. Acesso em 22 jul. 2013.

DIX, A., RODDEN, T., DAVIES, N., TREVOR, J., FRIDAY, A., PALFREYMAN, K.. Exploiting space and location as a design framework for interactive mobile systems. **Journal ACM Transactions on Computer-Human Interaction (TOCHI) -** Special issue on human-computer interaction with mobile systems, vol. 7, no. 3, p. 285-321, set. 2000.

ECLIPSE, Platform, Site oficial. Disponível em: <http://www.eclipse.org>. Acesso em: 05 jan. 2015.

FALK, . A., TAVARES, J. E. R., BARBOSA, J. L. V.. Tirésias: Um Modelo para Acessibilidade Ubíqua Orientado a Deficiência Visual. **Revista Brasileira de Computação Aplicada** (ISSN 2176-6649), Passo Fundo, v. 5, n. 1, p. 55-70, abr. 2013.

FARIAS, K., NUNES, I., SILVA V., LUCENA, C.. MAS-ML Tool: Um Ambiente de Modelagem de Sistemas Multi-Agentes. In: V Workshop on Software Engineering for Agent-oriented Systems, Fortaleza – CE, 2009. **Anais eletrônicos**… Fortaleza: SBC, 2009. Disponível em: <http://www.lbd.dcc.ufmg.br/bdbcomp/servlet/Evento?id=236>. Acesso em: 04 mai. 2013.

FARIAS, K., GARCIA, A., LUCENA, C.. Evaluating the Impact of Aspects on Inconsistency Detection Effort: A Controlled Experiment. OPUS Research Group/LES, Informatics Departament, PUC-Rio, 2012. In: Model Driven Engineering Languages and Systems, vol. 7590, 2012, p. 219-234. Disponível em: <http://link.springer.com/chapter/10.1007/978-3-642-33666-9_15#page-1>. Acesso em: 15 jul. 2013.

FOLDOC. Free On-line Dictionary of Computing. Disponível em: <http://foldoc.org/context>. Acesso em: 31 mai. 2014.

FOWLER, M. e SCOTTS, K.. UML Essencial – Um breve guia para a linguagem-padrão de modelagem de objetos. São Paulo: Bookman, vol. 2, 2000. p. 5-160.

GU, T., WANG, X. H., PUNG, H. K., ZHANG, D. Q.. An Ontology-based Context Model in Intelligent Environments. In: Proceedings of Communication Networks and Distributed Systems Modeling and Simulation Conference, San Diego, CA, USA, 2004. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.3.2194>. Acesso em: 14 jun. 2013.

HENRICKSEN, K., INDULSKA, J.. Modelling and Using Imperfect Context Information. In: Workshop Proceedings of the 2nd IEEE Conference on Pervasive Computing and Communications (PerCom2004) Orlando, FL, USA, March 2004. **Anais eletrônicos**… IEEE, 2004. Disponível em: <http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=8997>. Acesso em: 16 jun. 2014.

HOYOS, José R., GARCÍA-MOLINA, Jesus, BOTÍA, Juan A.. A domain-specific language for context modeling in context-aware systems**. The Journal of Systems and Software**, vol. 86, pg 2890–2905 em 2013.

JOST, Tiago A.. **Unipay – Um Modelo de Pagamento Móvel Voltado ao Comércio Ubíquo,** 2012. 123f. Dissertação de Mestrado do programa de pós-graduação em Computação Aplicada (PIPCA) da Universidade do Vale do Rio dos Sinos, São Leopoldo, RS, 2012.

KITCHENHAM, B., et al.: Evaluating Guidelines for Reporting Empirical Software Engineering Studies. In: Empirical Software Engineering, vol. 13, issue 1, fev. 2008. P. 97–112.

LANGE, C., CHAUDRON, M.: An Empirical Assessment of Completeness in UML Designs. In: 8th Empirical Assessment in Software Engineering, 2004, pp. 111–121, 2004. **Anais eletrônicos**… IET Digital Library. Disponível em: http://digital-library.theiet.org/content/conferences/10.1049/ic_20040404. Acesso em: 10 jan. 2015.

LANGE, C., CHAUDRON, M.: Effects of Defects in UML Models – An Experimental Investigation. In: International Conference on Software Engineering 2006, Shangai, China, p. 401–411, mai 2006. **Anais eletrônicos**… New York: ACM DL, 2007. Disponível em: <http://dl.acm.org/citation.cfm?id=1134341>. Acesso em: 22 mai. 2013.

LARMAN, Craig. Utilizando UML e Padrões – Uma Introdução à Análise e ao Projeto Orientados à Objetos. Porto Alegre, Bookman, 2000. p. 10-696.

LOPES, João, GUSMÃO, Márcia, Duarte Cauê, DAVET, Patricia, SOUZA, Rodrigo, PERNAS, Ana, YAMIN, Adenauer, GEYER Cláudio. Toward a distributed architecture for context awareness in ubiquitous computing. **Journal of Applied Computing Research**, vol. 3, nro 1, p. 19-33, jun. 2013.

MANNADE, Rahul B. BHANDE, Amol B.. Challenges of Mobile Computing: An Overview. **International Journal of Advanced Research in Computer and Communication Engineering**. vol. 2, issue 8, p. 3109-3114, aug. 2013.

MARTIN, Robert. Agile Software Development, Principles, Patterns, and Practices, Pearson. Education, out. 2002. p. 101-125.

NADOVEZA, Drazen, KIRITSIS, Dimitris. Ontology-based approach for context modeling in enterprise applications. In: Elsevier Science Ltd - Computer in Industry, vol. 65, issue 9, p. 1218–1231, dec. 2014.

NADOVEZA, Drazen, KIRITSIS, Dimitris. Concept for context-aware manufacturing dashboard applications. In: Manufacturing Modelling, Management, and Control, vol. 7, p. 204-209, 2013. Disponível em: <http://www.ifac-papersonline.net/Detailed/59881.html>. Acesso em: 22 jun. 2013.

NOVA Júnior, H. S. V.. **Proposta de um Sistema Ubíquo para Ambiente Hospitalar de Urgência**. 2008. 70 f. Dissertação de Mestrado do programa de pós-graduação em Computação da Universidade Federal de Pernambuco, Pernambuco, 2013.

ROEHRS, A.. **4iPay: Modelo para Sistemas de Pagamento Móvel em Comércio Ubíquo**. 2012. 100 f. Dissertação de Mestrado do programa de pós-graduação em Computação Aplicada (PIPCA) da Universidade do Vale do Rio dos Sinos, São Leopoldo, RS, 2012.

SATYANARAYANAN, Mahadev. Mobile Computing: the Next Decade**.** School of Computer Science Carnegie Mellon University - Primeira Oficina ACM no Mobile Cloud Computing & Serviços. In: ACM SIGMOBILE Mobile Computing and Communications Review, vol. 15, issue2, p. 2-10 jun. 2010.

SCHMIDT, A., BEIGL, M., GELLERSEN H.. There is more to context than location**.** In: Elsevier Science Ltd - Computers & Graphics, vol. 23, issue 6, p 893–901, dec. 1999.

SERRAL, Estefânia; VALDERAS, Pedro; PELECHANO, Vicente. Towards the Model Driven Development of Context-Aware Pervasive Systems. In: Pervasive and Mobile Computing, vol 6, issue 2, pg 254–280, abr. 2010.

SILVA, Viviane Torres, CHOREN, Ricardo, LUCENA, Carlos J.P. (2008). MAS-ML: A Multiagent System Modelling Language. Departamento de Sistemas Informáticos Y Computación Universidad Completense de Madrid. **Jornal: Agent-Oriented Software Engineering**, vol.2, nro. 4, 2008.

SINDICO, A., GRASSI, V.. Model driven development of context aware software systems. In: International Workshop on Context-Oriented Programming (COP) Genova, Italy, pp.1-5, jul. 2009

STRANG, T., LINNHOFF-POPIEN, C.. A Context Modeling Survey. In: International Workshop on Advanced Context Modelling, Reasoning And Management at UbiComp, Nottingham, England, set. 2004. Disponível em: <http://elib.dlr.de/7444/>. Acesso em: 22 jul. 2013.

SILVA, Carlos Vinícius Pereira; CAMPOS, Danylo de Castro Campo; MOURA, Déborah Carvalho; NERY, Paulo. GQM: Goal – Question – Metric. In: Seminário de TI do CIn-UFPE, 14 ago. 2009.

SUN, Wuliang, FRANCE, Robert B., RAY, Indrakshi. Contract-Aware Slicing of UML Class Models. In: Springer-Verlag Berlin Heidelberg 2013: A. Moreira et al. (Eds.): MODELS 2013, LNCS 8107, pp. 724–739, 2013.

TAVARES, J. E. R.. **Hefestos – Um Modelo para Suporte à Acessibilidade Ubíqua**. 2011. 111 f. Dissertação de Mestrado do programa de pós-graduação em Computação Aplicada (PIPCA) da Universidade do Vale do Rio dos Sinos, São Leopoldo, RS, 2011.

UML. Unified Modeling Language. Disponível em: <http://www.uml.org>. Acesso em: 12 ago. 2014.

UMLBase – Expand your UML Knowledge and get certification (2014) Disponível em: <http://umlbase.com/learn/fundamentals/the-uml-metamodel/>. Acesso em: 10 mai, 2014.

WAISER, Mark. The Computer for the 21st Century. In: Scientific American Special Issue on Communications. Computers, and Networks, set. 1991.

WOHLIN, et al. Experimentation in Software Engineering: an Introduction. Kluwer Academic Publishers, Springer. New York, 2000.

ANEXO I

Introdução aos Questionários

Nome (opcional):			
Idade:		**Sexo:**	( ) F ( ) M
Profissão:		**Empresa em que trabalha:**	
Cargo atual:		**Quanto tempo está neste cargo:**	

Maior grau de escolaridade:		Técnico	**Qual sua formação acadêmica:**		Sistemas de Informação
		Graduação			Ciências da Computação
		Mestrado			Engenharia da Computação
		Doutorado			Análise de Sistemas
		Outra. Qual ?			Outro: Qual ?

Por quanto tempo você estudou (tem estudado) em universidades?		Menos de 2 anos	**Se cargo atual se encaixa mais em qual especialidade:**		Programador
		De 2 a 4 anos			Analista
		De 5 a 6 anos			Arquiteto
		De 7 a 8 anos			Gerente
		Mais de 8 anos			Outro: Qual ?

Quanto tempo tem de experiência em desenvolvimento de software ?		Menos de 2 anos	**Quanto tempo tem de experiência em modelagem de software ?**		Menos de 2 anos
		De 2 a 4 anos			De 2 a 4 anos
		De 5 a 6 anos			De 5 a 6 anos
		De 7 a 8 anos			De 7 a 8 anos
		Mais de 8 anos			Mais de 8 anos

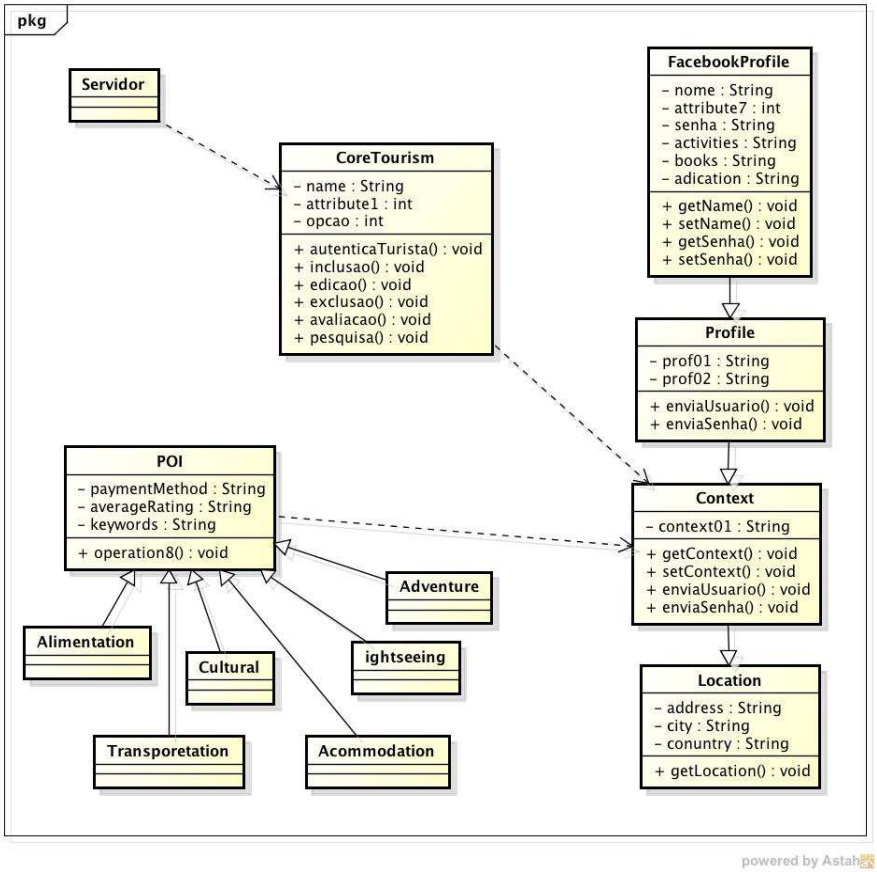
Este questionário tem o objetivo de avaliar a metodologia de modelagem proposta pela dissertação da Vivian Pedó. Desse modo, as respostas para as questões abaixo devem ser baseadas na experiência do participante. O questionário possui duas partes, uma para caracterizar o participante e a outra para coletar as informações. Os participantes não estão sendo avaliados e seus dados não serão divulgados. As questões pessoais acima servem somente para dividirmos conforme a categoria de acordo com a metodologia.

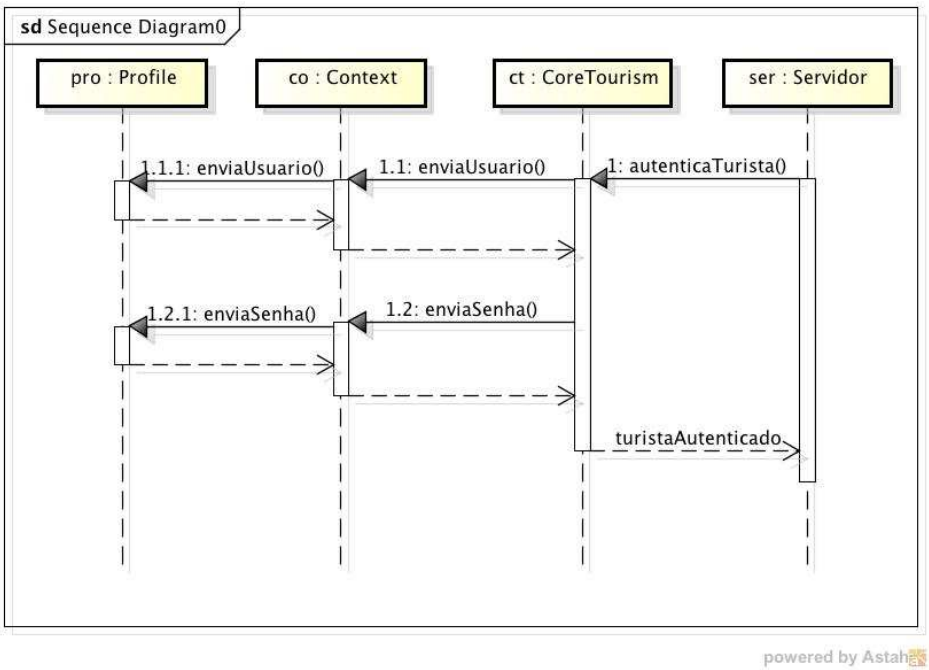
Muito obrigada pela sua participação !

Questionário 1

Tempo:	Hora de Início: ___:___	Hora de Término: ___:___

Questão 01 - Aplicação UbiTour





Questão 01 - Aplicação UbiTour

Questão 01: Suponha que você é um desenvolvedor do projeto UbiTour, um software de turismo ubíquo. Considerando o diagrama de classes e sequência acima, como você implementaria a classe **Core Tourism**?

A) ( )
```
class CoreTourism {
  public void autenticaTurista(){
    pro.enviaUsuario();
    co.enviaUsuario();
    pro.enviaSenha();
    servidor = ct.turistaAutenticado()
}
```

B) ( )
```
class CoreTourism {
  public void autenticaTurista(){
    co.enviaUsuario();
    co.enviaSenha();
  }
}
```

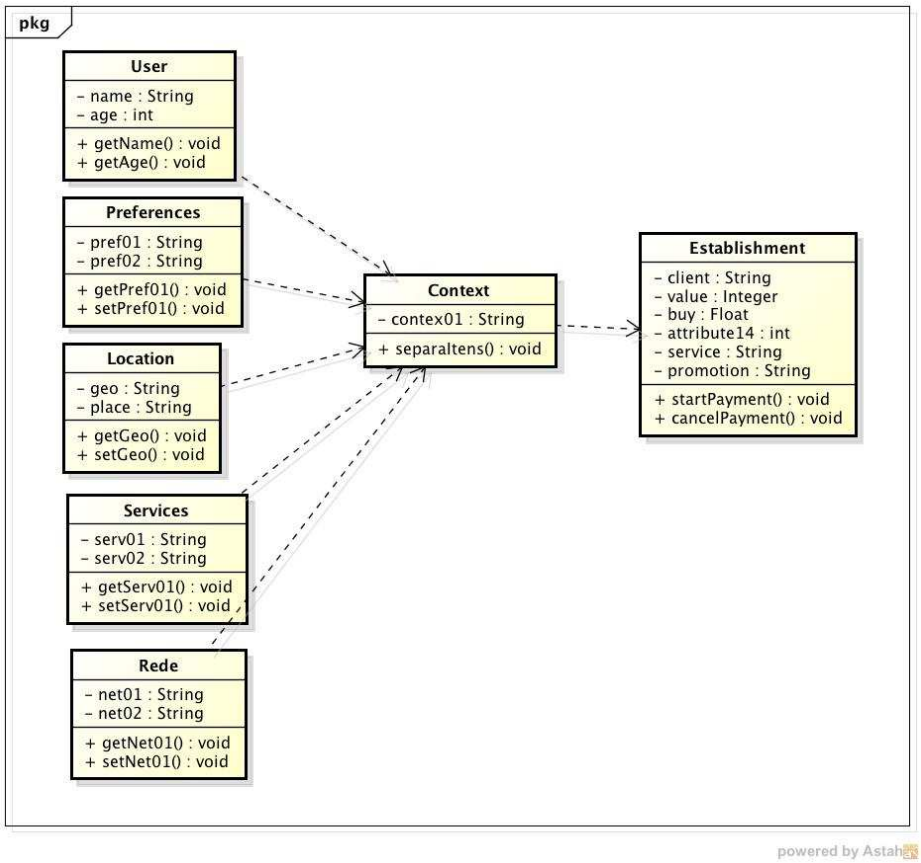
C) ( )
```
class CoreTourism {
  public void autenticaTurista(){
    co.enviaUsuario();
    co.enviaSenha();
    pro.enviaSenha();
    context = co.turistaAutenticado()
}
```

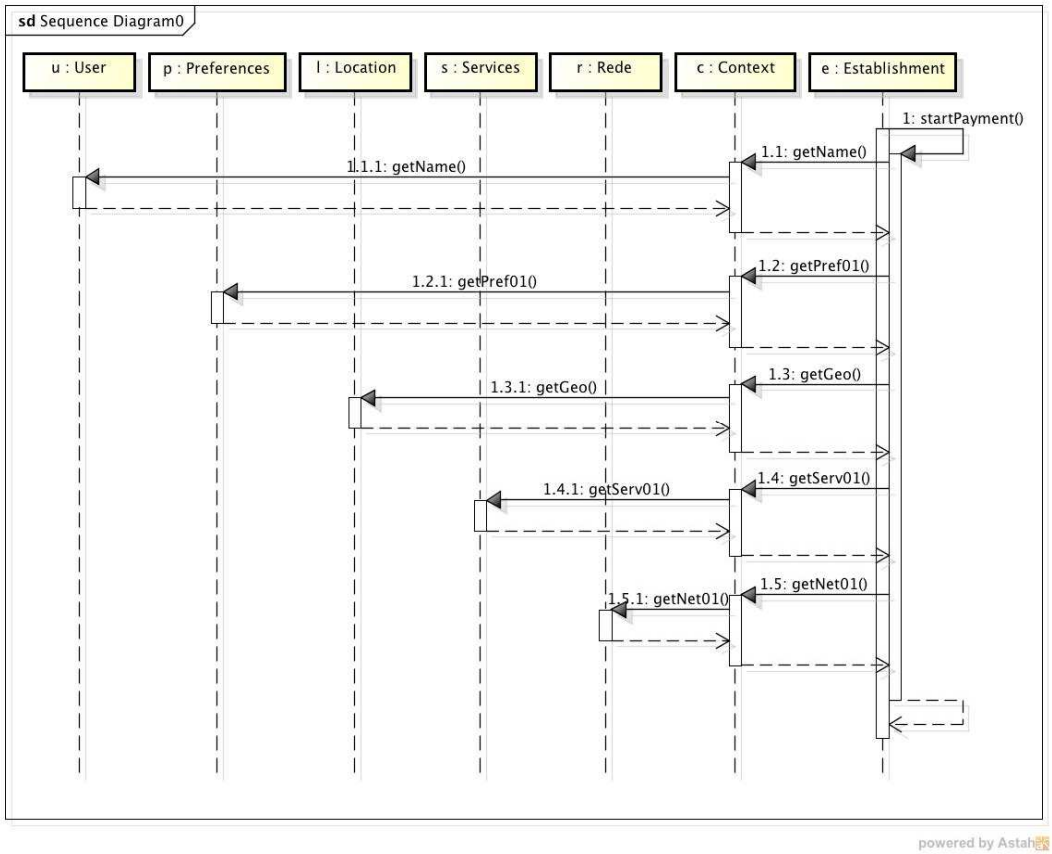
D) ( )
```
class CoreTourism {
  public void autenticaTurista(){
    pro.enviaUsuario();
    pro.enviaSenha();
    }
}
```

E) ( ) Nenhuma interpretação pode ser feita porque existe problema no modelo.

Observações da questão 1

Questão 02 - Aplicação 4iPay





Questão 02 - Aplicação 4iPay

Questão 02: Suponha que você é um desenvolvedor do 4iPay. Considerando o diagrama de classe e sequência acima que esboça as consultas feitas ao efetuar um pagamento, como você implementaria as **classes do contexto**?

A) ( )
```
public class Context extends User { }
public class Context extends Preferences { }
public class Context extends Location {  }
public class Context extends Services {  }
public class Context extends Rede {  }
```

B) ( )
```
public class Context {
   private User us;
   private Location loc;
   private Preferences pre;
   private Services ser;
   private Rede red;
}
class class User {...}
class Preferences {...}
class class Location {...}
class Services {...}
class Rede {...}
```

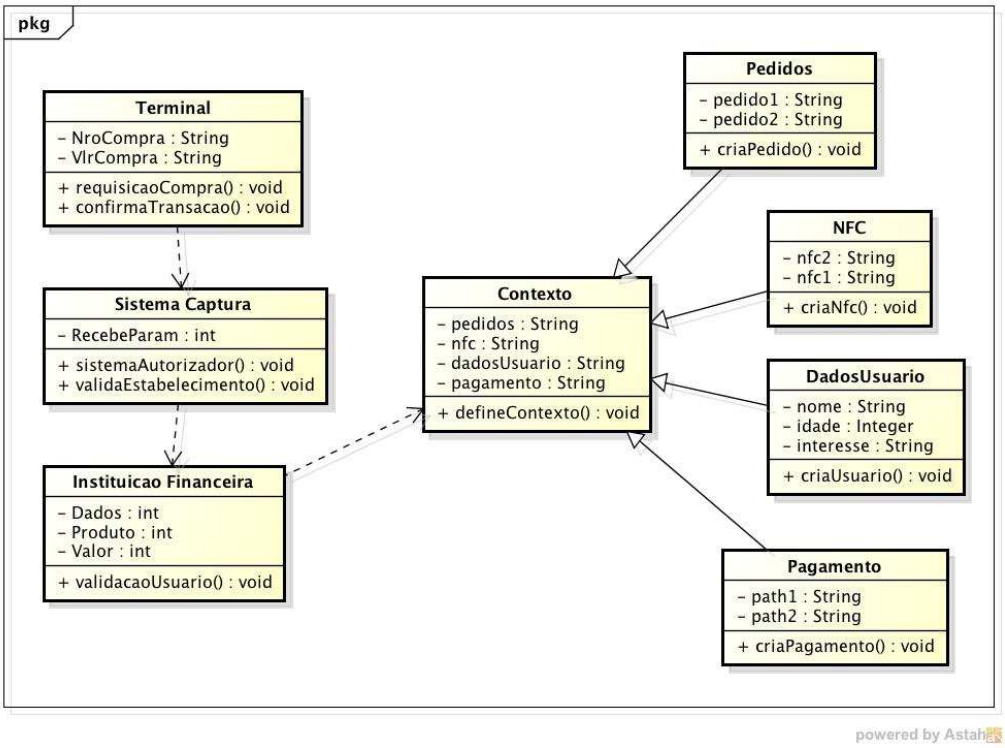
C) ( )
```
public class Context {
  public User us;
  public Preferences pre;
  public Location loc;
  public Services ser;
  public Rede red;
}
class class User {...}
class class Preferences {...}
class Location {...}
class Services {...}
class Rede {...}
```

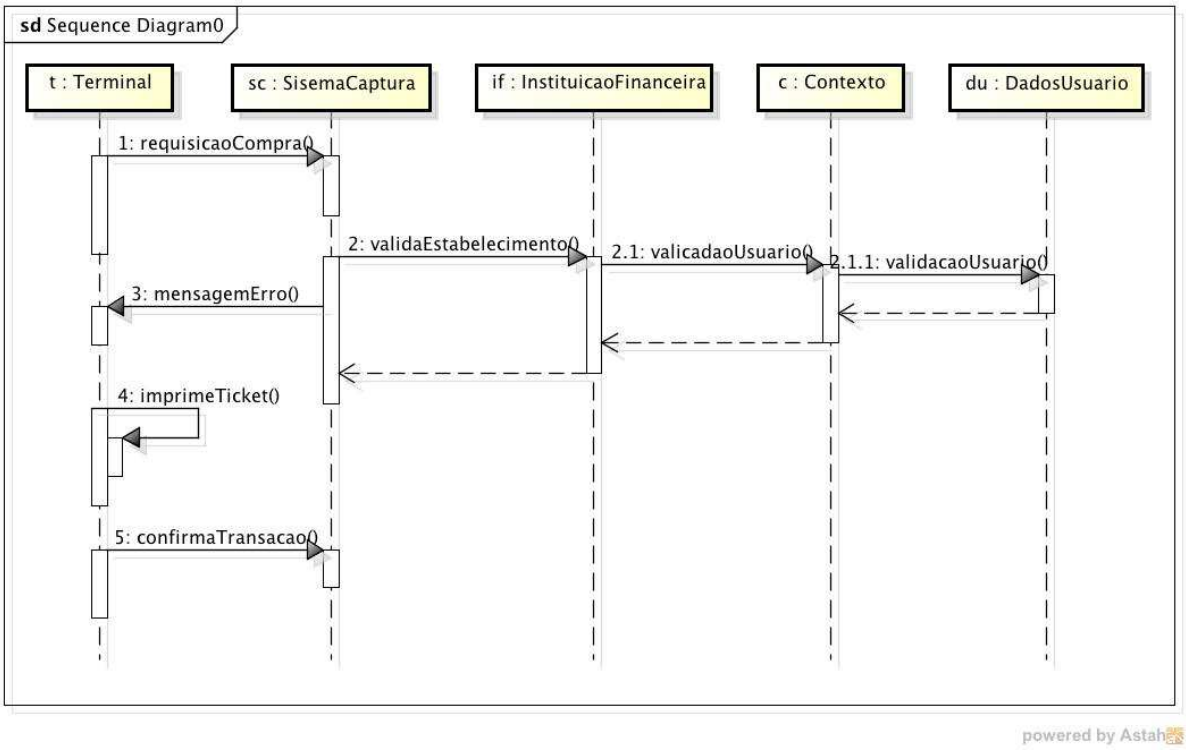
D) ( )
```
public class Context {
  public class extends User {  }
  public class extends Preferences {   }
  public class extends Location {  }
  public class extends Services {  }
  public class extends Rede {   }
}
```

E) ( ) Nenhuma interpretação pode ser feita porque existe problema no modelo.

Observações da questão 2

Questão 03 - Aplicação UniPay





Questão 03 - Aplicação UniPay

Questão 03: Suponha que você é um desenvolvedor do projeto de pagamento móvel UniPay. Considerando o diagrama de classe, diagrama de sequência acima analise o código Java abaixo onde foi implementada as classes e marque como você implementaria uma chamada na classe **Context** para recuperar a nome do usuário da classe **DadosUsuario** ?

A) ( )
```
public class Contexto {
  private DadosUsuario du;
  private Pedidos ped;
  private NFC nfc;
  private Pagamento pag;
}
public Class Context{
  Public void defineContexto (){
     du.validacaoUsuario.Idade }
}
```

B) ( )
```
public class Contexto {
   private DadosUsuario du;
   private Pedidos ped;
   private NFC nfc;
   private Pagamento pag;
}
public Class InsituicaoFinanceira{
   Public void validacaoUsuario (){
      user.validacaoUsuario.Idade; }
}
```

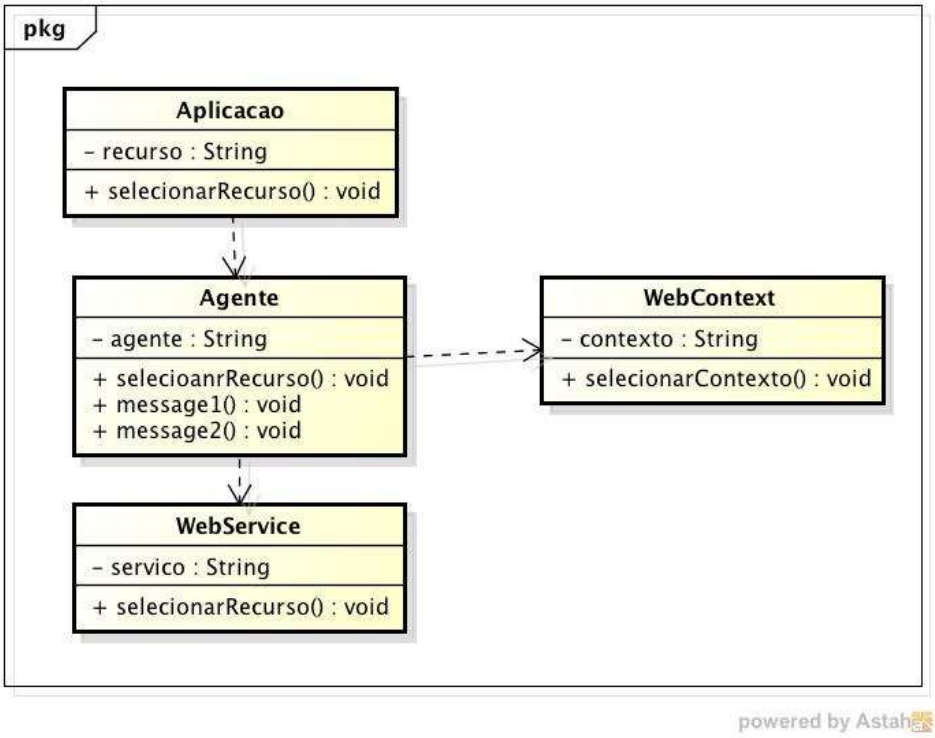
C) ( )
```
public class Context0 {
   private DadosUsuario user;
   private Pedidos ped;
   private NFC nfc;
   private Pagamento pag;
}
public Class InsituicaoFinanceira{
   Public void defineContexto (){
      user.validacaoUsuario(Idade);}
}
```

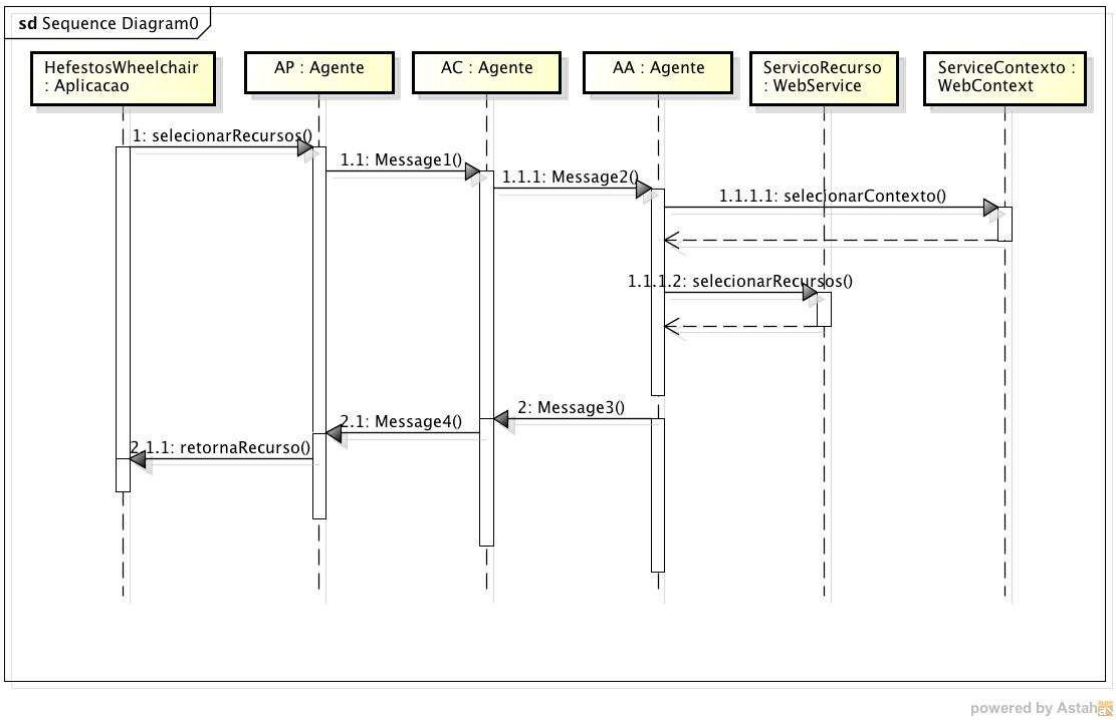
D) ( )
```
public class Contexto {
  private DadosUsuario user;
  private Pedidos ped;
  private NFC nfc;
  private Pagamento pag;
}
public Class Contexto{
   Public void validacaoUsuario (){
     user.validacaoUsuario(Idade); }
}
```

E) ( ) Nenhuma interpretação pode ser feita porque existe problema no modelo.

Observações da questão 3

Questão 04 - Aplicação Hefestos





Questão 04 - Aplicação Hefestos

Questão 04: Suponha que você é um desenvolvedor de uma aplicação de acessibilidade para pessoas com deficiência motora nas pernas. Considerando o diagrama de classe e sequência acima, como você implementaria a classe **Agente** tipo da instância **AA** ?

A) ( )
```
class AA {
 public void message2(){
   serviceContexto.selecionarContexto();
   servicoRecurso.selecionarRecursos();
   ap.retornaRecurso();
   }
}
```

B) ( )
```
class AA {
  public void selecionarRecursos(){
    serviceContexto.selecionarContexto();
    servicoRecurso.selecionarRecursos();
  }
}
```

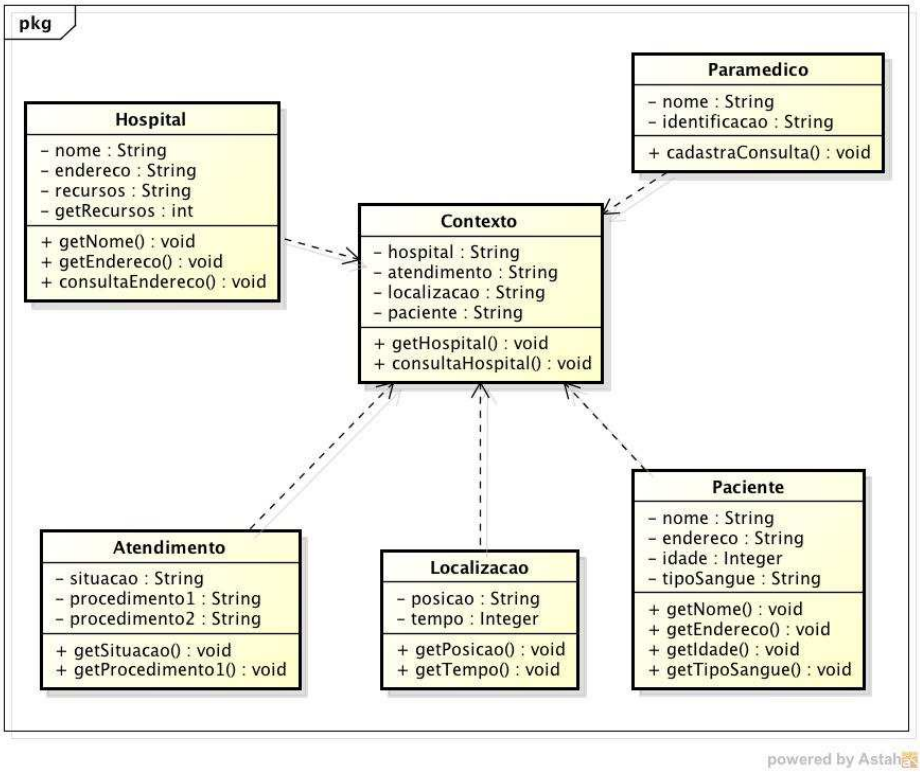
C) ( )
```
class AA {
  public void message2(){
    aa.message2();
    serviceContexto.selecionarContexto();
    servicoRecurso.selecionarRecursos();
  }
}
```

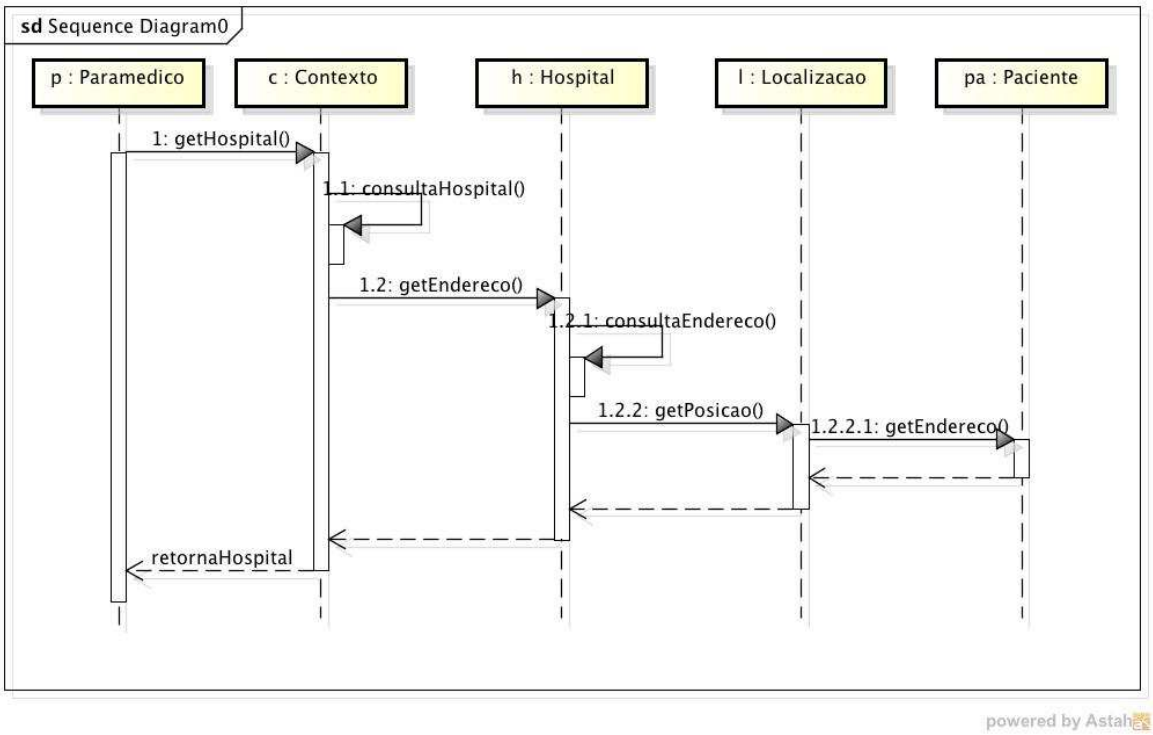
D) ( )
```
class AA {
  public void message2(){
    serviceContexto.selecionarContexto();
    servicoRecurso.selecionarRecursos();
  }
}
```

E) ( ) Nenhuma interpretação pode ser feita porque existe problema no modelo.

Observações da questão 4

Questão 05 - Aplicação SAUM





Questão 05 - Aplicação SAUM

Questão 05: Supondo que você é um desenvolvedor de uma aplicação para atendimento de paramédicos e localização e entrada no hospital durante o deslocamento da ambulância. Considerando o diagrama de classe e sequência acima, como você implementaria a classe **Contexto**?

A) ( )
```
class Contexto {
  public void getHospital(){
    c.consultaHospital();
    h.getEndereco();
    h.consultaEndereco();
  }
}
```

B) ( )
```
class Contexto {
   public void getHospital(){
     c.consultaHospital();
     c.getEndereco();
   }
}
```

C) ( )
```
class Contexto {
  public void getHospital(){
    c.consultaHospital();
    h.getEndereco();
  }
}
```

D) ( )
```
class Contexto {
  public void getHospital(){
    c.consultaHospital();
    c.getEndereco();
    h.consultaEndereco();
  }
}
```

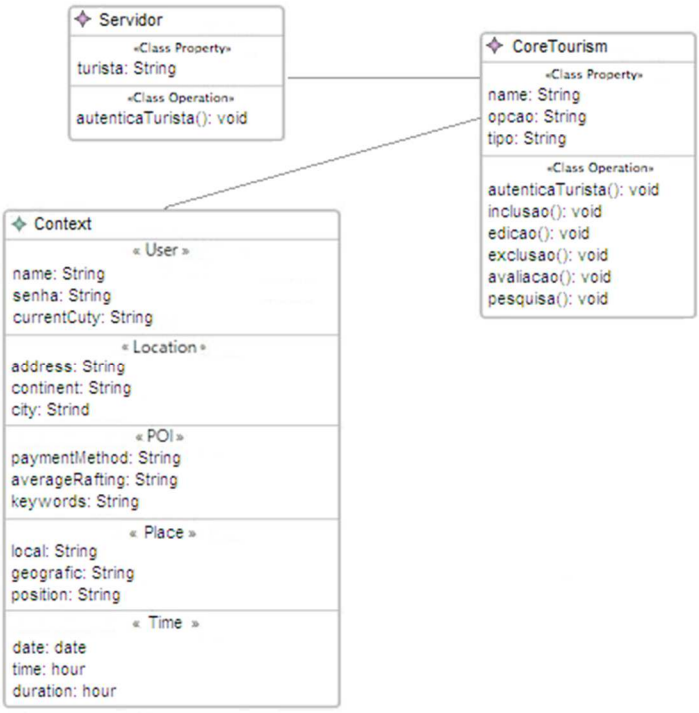
E) ( ) Nenhuma interpretação pode ser feita porque existe problema no modelo.

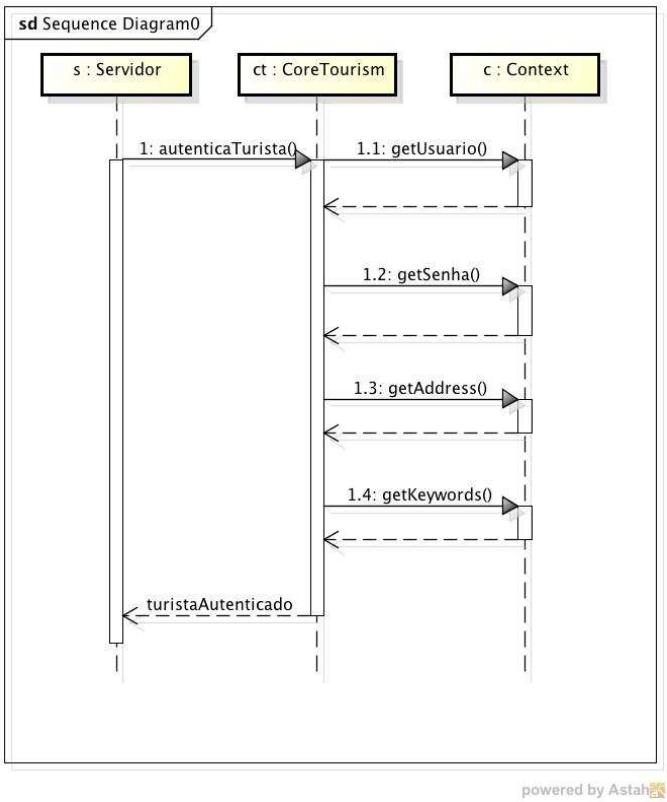
Observações da questão 5

Questionário 2

Tempo:	Hora de Início: ___:___	Hora de Término: ___:___

Questão 01 - Aplicação UbiTour





Questão 01 - Aplicação UbiTour

Questão 01: Suponha que você é um desenvolvedor do projeto UbiTour, um software de turismo ubíquo. Considerando o diagrama de classes e sequência acima, como você implementaria **a classe Context** ?

A) ( )
```
public class Context{
    Servidor[] serv;
    User[] use;
    Location[] loc;
    POI[] poi;
    Place[] plc;
    Time[] tme;
}
```

B) ( )
```
public class Context{
    Servidor[] serv;
    coreTourism[] cto;
    User[] use;
    Location[] loc;
    POI[] poi;
    Place[] plc;
    Time[] tme;
}
```

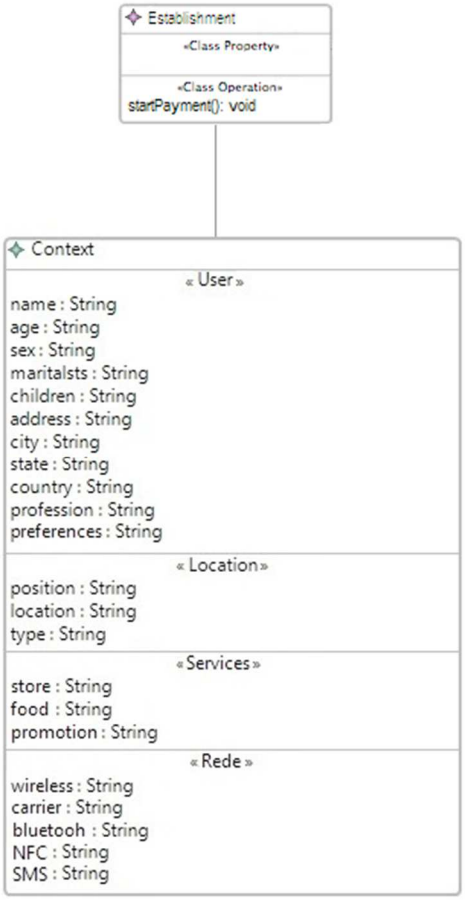
C) ( )
```
public class Context{
    User[] use;
    Location[] loc;
    POI[] poi;
    Place[] plc;
    Time[] tme;
}
```

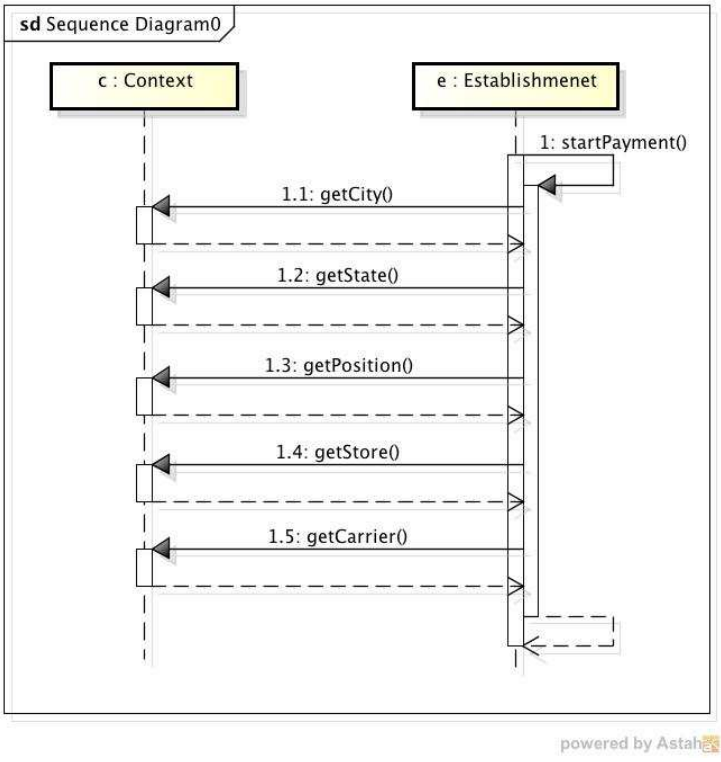
D) ( )
```
public class Context{
    User[] use;
    Location[] loc;
    Place[] plc;
    Time[] tme;
    CoreTourism[] cto;
}
```

E) ( ) Nenhuma interpretação pode ser feita porque existe problema no modelo.

Observações da questão 1

Questão 02 - Aplicação 4iPay





Questão 02 - Aplicação 4iPay

Questão 02: Suponha que você é um desenvolvedor do 4iPay. Considerando o diagrama de classe e sequência acima que esboça as consultas feitas ao efetuar um pagamento, como você implementaria a classe **Establishment**?

A) ( )
```
class Establishment {
  public void startPayment(){
    c.getCarrier();
    c.getCity();
    c.getStore();
    c.getState();
    c.getPosition();}
}
```

B) ( )
```
class Establishment {
  public void startPayment(){
    c.getCity();
    c.getState();
    c.getPosition();
    c.getStore();
    c.getCarrier();}
}
```

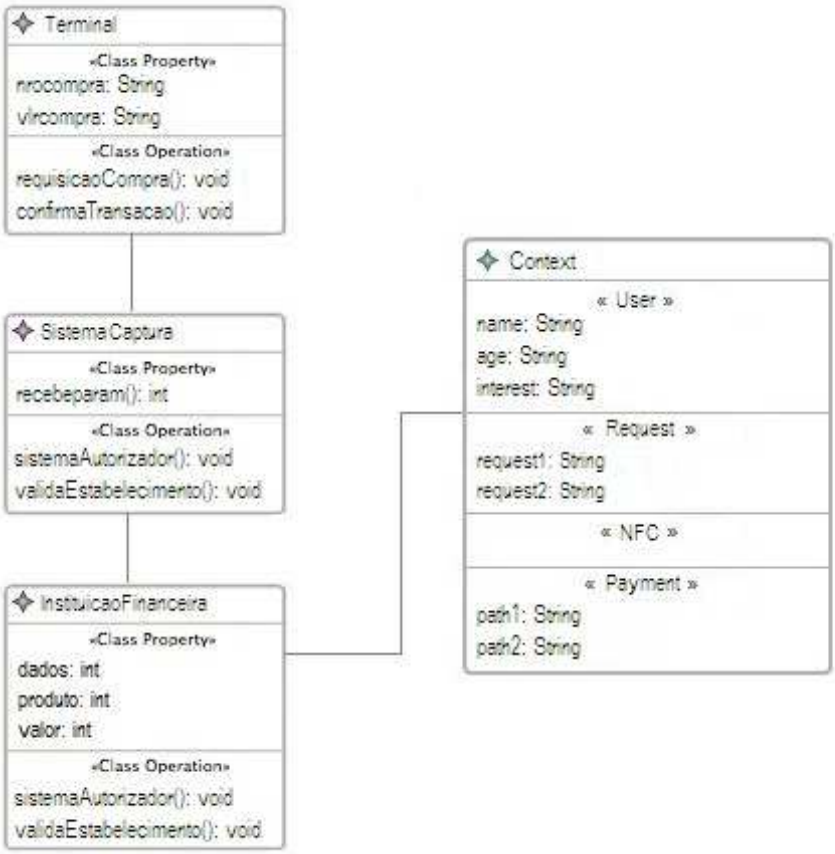
C) ( )
```
class Establishment {
  public void startPayment(){
    c.getAddress();
    c.getState();
    c.getPosition();
    c.getStore();
    c.getCarrier();}
}
```

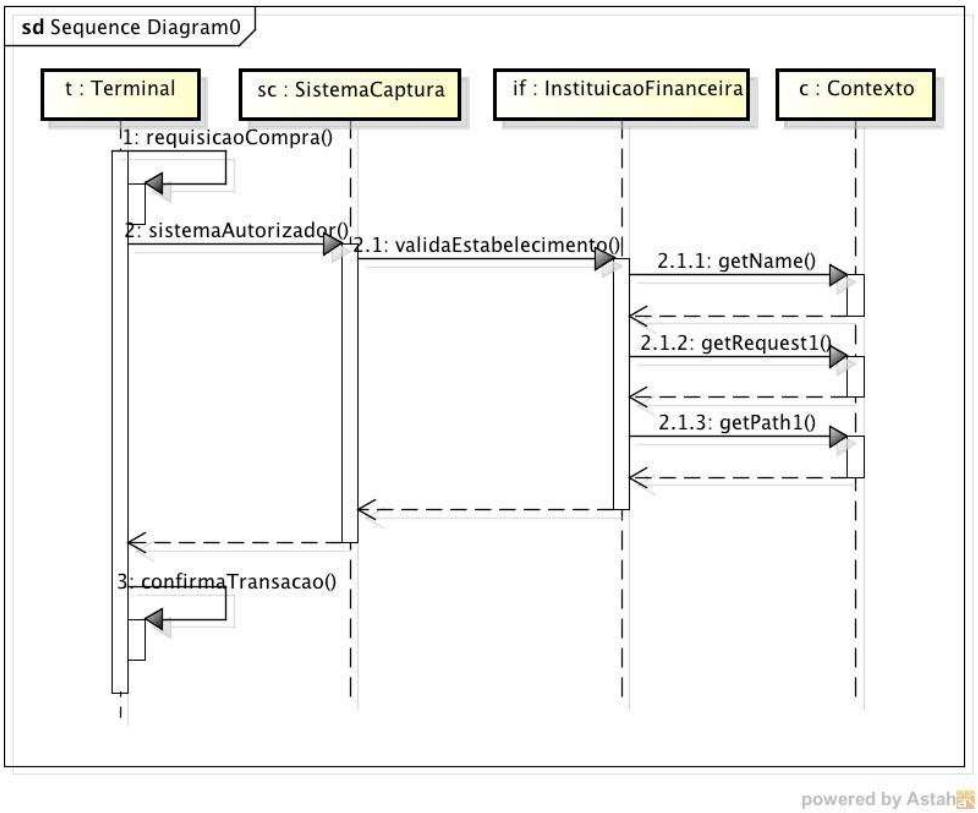
D) ( )
```
class Establishment {
  public void startPayment(){
    c.getCity();
    c.getState();
    c.getPosition();
    c.getStore();
    c.getCarrier();
    establishment = Context(autenticado);}
}
```

E) ( ) Nenhuma interpretação pode ser feita porque existe problema no modelo.

Observações da questão 2

Questão 03 - Aplicação UniPay





Questão 03 - Aplicação UniPay

Questão 03: Suponha que você é um desenvolvedor do projeto de pagamento móvel UniPay. Considerando o diagrama de classe, diagrama de sequência acima como você implementaria a definição da classe **Instituição Financeira** ?

A) ( )
```
class InstituicaoFinanceira {
  public void sistemaautorizador ()
   {
    c.getName();
    c.getRequest1();
    c.getPath1();
    }
}
```

B) ( )
```
class InstituicaoFinanceira {
  public void validaestabelecimento ()
    {
    if.getName();
    if.getRequest1();
    if.getPath1();
    }
}
```

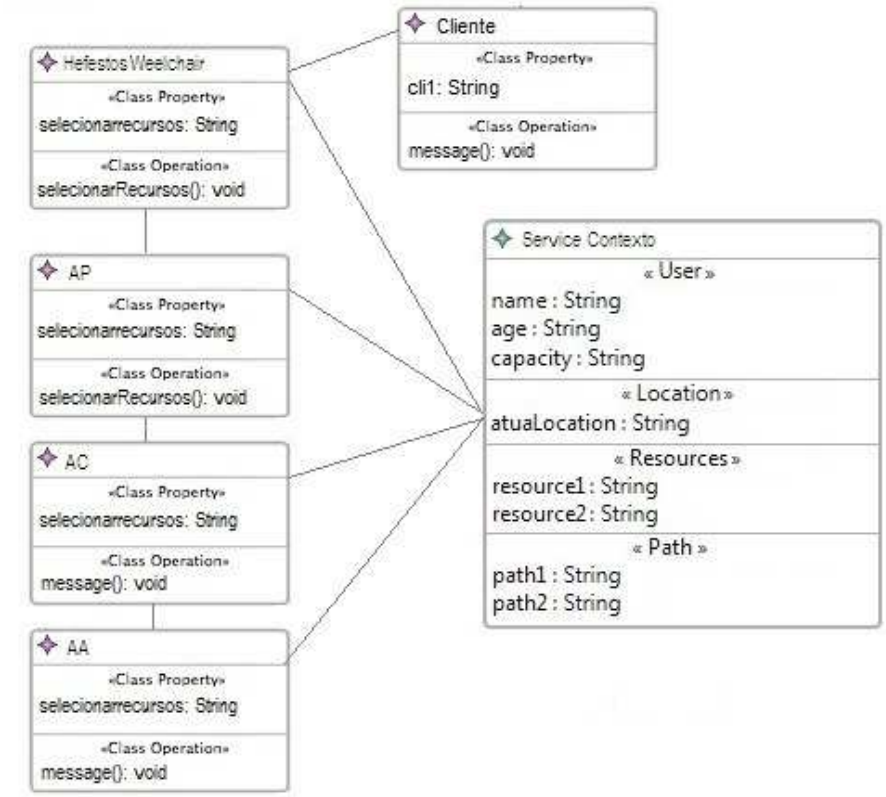
C) ( )
```
class InstituicaoFinanceira {
  public void validaestabelecimento ()
    {
    c.getName();
    c.getRequest1();
    c.getPath1();
    }
}
```

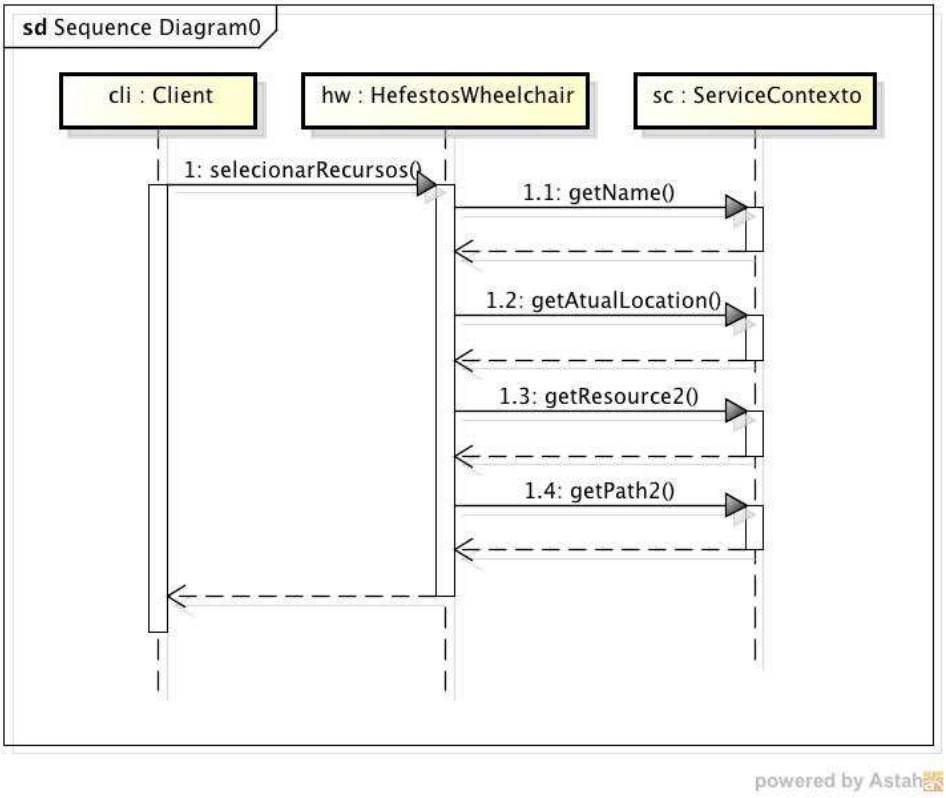
D) ( )
```
class InstituicaoFinanceira {
  public void sistemaautorizador ()
    {
    if.getPath1();
    if.getName();
    if.getRequest1();
    }
}
```

E) ( ) Nenhuma interpretação pode ser feita porque existe problema no modelo.

Observações da questão 3

Questão 04 - Aplicação Hefestos





Questão 04 - Aplicação Hefestos

Questão 04: Suponha que você é um desenvolvedor de uma aplicação de acessibilidade para pessoas com deficiência motora nas pernas. Considerando o diagrama de classe e sequência acima, como você implementaria todos os métodos das classes conforme o diagrama de sequência a partir da classe ***Hefestos Weelchair***?

A) ( )
```
Private class HefestosWeelchair
{
  public void selecionarRecursos (){
    sc.Path2();
    sc.getName();
    sc.Resource2();
    sc.getAtualLocation();}
}
```

B) ( )
```
Private class HefestosWeelchair
{
  public void selecionarRecursos (){
    c.getName();
    c.getAtualLocation();
    c.Resource1();
    c.Path1();}
}
```

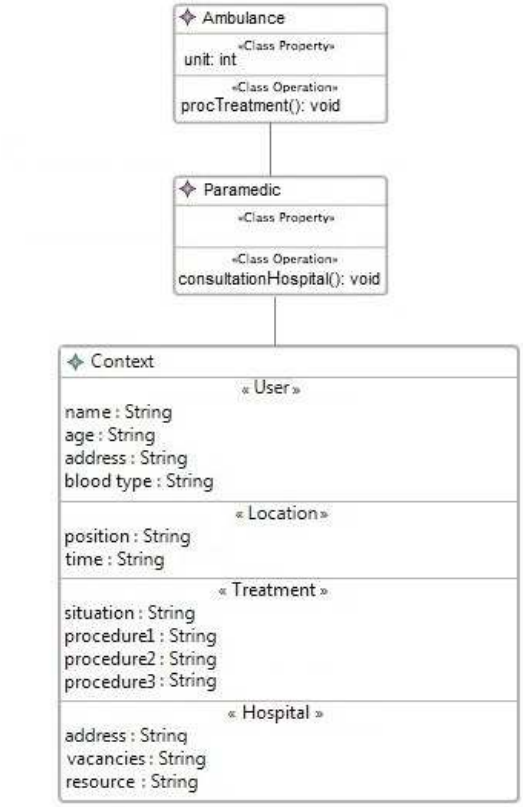
C) ( )
```
Private class HefestosWeelchair
{
  public void selecionarRecursos (){
    hw.getName();
    hw.getAtualLocation();
    hw.Resource2();
    hw.Path2();}
}
```

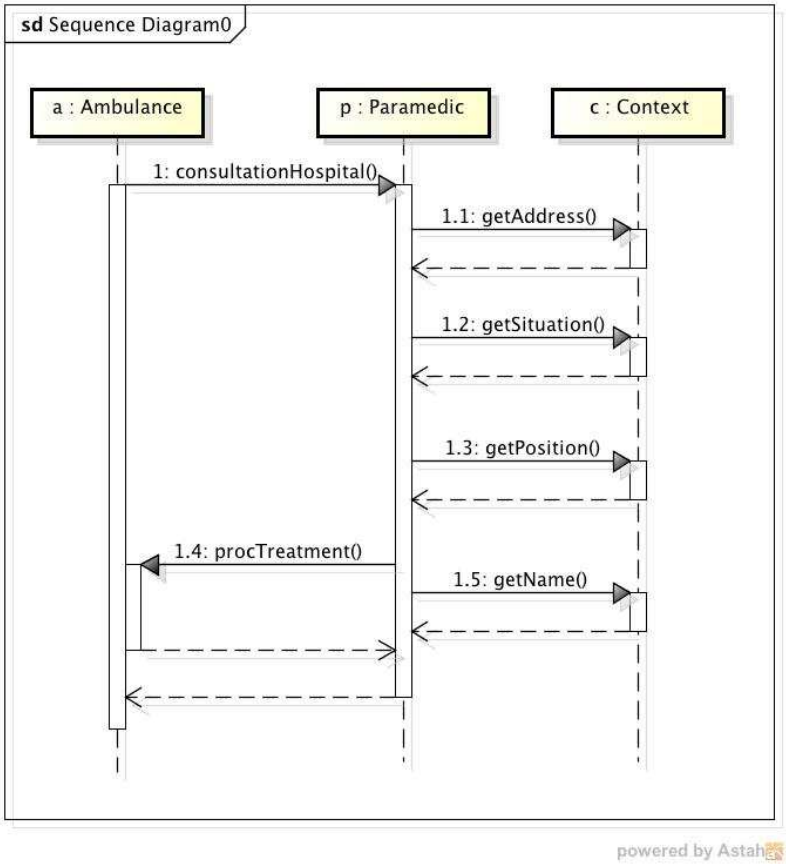
D) ( )
```
Private class HefestosWeelchair
{
  public void selecionarRecursos (){
    sc.getName();
    sc.getAtualLocation();
    sc.Resource2();
    sc.Path2();}
}
```

E) ( ) Nenhuma interpretação pode ser feita porque existe problema no modelo.

Observações da questão 4

Questão 05 - Aplicação SAUM





Questão 05 - Aplicação SAUM

Questão 05: Suponha que você é um desenvolvedor de uma aplicação para uma instituição financeira. Considerando o diagrama de classe e sequência acima, como você implementaria a classe ***Paramedic***?

A) ( )
```
public class Paramedic {
   public void consultationHospital(){
      c.getAddress();
      c.getSituation();
      c.getPosition();
      a.procTreatment();
      c.getName();}
}
```

B) ( )
```
public class Paramedic {
   public void consultationHospital(){
      a.consultationHospital();
      c.getAddress();
      c.getSituation();
      c.getPosition();
      a.procTreatment();
      c.getName();}
}
```

C) ( )
```
public class Paramedic {
   public void consultationHospital(){
      c.getName();
      c.getAddress();
      c.getSituation();
      c.getPosition();
      p.procTreatment();}
}
```

D) ( )
```
public class Paramedic {
   public void consultationHospital(){
      p.procTreatment();
      c.getName();
      c.getAddress();
      c.getSituation();
      c.getPosition();}
}
```

E) ( ) Nenhuma interpretação pode ser feita porque existe problema no modelo.

Observações da questão 5