UNIVERSIDADE FEDERAL DO MARANHÃO
CENTRO DE CIÊNCIA EXATAS E TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE ELETRICIDADE

DÉBORA RODRIGUES STEFANELLO

UM *FRAMEWORK* BASEADO EM MDE E *WEAVING* PARA SUPORTE AO DESENVOLVIMENTO DE SISTEMAS DE *SOFTWARE* SENSÍVEIS AO CONTEXTO

SÃO LUÍS - MA
2017

UNIVERSIDADE FEDERAL DO MARANHÃO
CENTRO DE CIÊNCIA EXATAS E TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE ELETRICIDADE

UM *FRAMEWORK* BASEADO EM MDE E *WEAVING* PARA SUPORTE AO DESENVOLVIMENTO DE SISTEMAS DE *SOFTWARE* SENSÍVEIS AO CONTEXTO

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia de Eletricidade da Universidade Federal do Maranhão, para obtenção do título de mestre em Engenharia de Eletricidade - Área de Concentração: Ciência da Computação.

Orientador: Dr. Denivaldo Cicero Pavão Lopes

SÃO LUÍS - MA
2017

UM *FRAMEWORK* BASEADO EM MDE E *WEAVING* PARA SUPORTE AO DESENVOLVIMENTO DE SISTEMAS DE *SOFTWARE* SENSÍVEIS AO CONTEXTO

DÉBORA RODRIGUES STEFANELLO

Aprovada em 25 de Janeiro de 2017

BANCA EXAMINADORA

Prof. Denivaldo Cícero Pavão Lopes, Dr.
(Orientador)

Profa. Eveline de Jesus Viana Sá, Dra.
(Membro da Banca Examinadora)

Prof. Vicente Leonardo Paucar Casas,Dr.
(Membro da Banca Examinadora)

RESUMO

Nos últimos anos, algumas pesquisas têm levado em conta as problemáticas relacionadas ao desenvolvimento de sistema de software, em especial, aqueles que fazem o uso de contexto, i.e. *context-aware systems*. Nesta dissertação de mestrado, aborda-se a complexidade no desenvolvimento de *context-aware system*. A solução proposta visa fornecer suporte para o desenvolvimento de *context-aware system* através de um *framework* baseado em MDE e *Weaving*. Por um lado, MDE permite a utilização de modelos para gerenciar a complexidade no desenvolvimento de *sofware*, enquanto a técnica de *weaving* suporta a criação de inter-relacionamentos entre elementos de modelos diferentes, mas complementares. A técnica de *weaving* é utilizada para criar um modelo de *weaving* que estabelece inter-relacionamentos entre os elementos de um PIM (*Platform Independent Mode*l) e de vários PDMs (*Platform Description Model*). Uma implementação do *framework* proposto e um exemplo ilustrativo ajudam a entender a proposta e mostram a sua viabilidade. Um comparativo entre a solução proposta e os trabalhos encontrados na literatura é feito, mostrando os pontos negativos e positivos da solução proposta.

Palavras Chave: Engenharia Dirigida por Modelos, *Framework*, Sistemas sensíveis ao contexto, *Weaving*.

ABSTRACT

In recent years, some research has taken into account problems related to the development of software systems, especially those that make use of context, i.e. context-aware systems. In this dissertation the complexity of context-aware system development is discussed. The proposed solution aims to provide support for the context-aware system development through a framework based on MDE and Weaving. On the one hand, MDE allows the use of models to manage complexity in software development, while the weaving technique supports the creation of interrelations between elements of different but complementary models. The weaving technique is used to create a weaving model that establishes interrelationships between the elements of a Platform Independent Model (PIM) and several Platform Description Model (PDMs). An implementation of the proposed framework and an illustrative example help to understand the proposal and its feasibility. A comparison between the proposed solution and the works found in the literature Done, showing the negative and positive points of the proposed solution..

KeyWords: Model Driven Engineering, Framework, Context-aware Systems, Weaving.

Dedicatória:

Dedico este trabalho aos meus pais

Luiz Arcangelo Pegoraro Stefanello

e Creusenir Furtado Rodrigues Stefanello

Agradecimentos

Agradeço em primeiro lugar a Deus por sua infinita bondade.
Aos meus pais Luiz Arcangelo Pegoraro Stefanello e Creusenir Furtado Rodrigues Stefanello pelo apoio, incentivo e força para a conclusão deste desafio, sem vocês eu jamais conseguiria.
Ao meu Orientador Dr. Denivaldo Cicero Pavão Lopes por todo o apoio oferecido, paciência, incentivo e toda a orientação necessária para conclusão desta pesquisa e principalmente por acreditar em mim.
Ao professor PhD. Zair Abdelouahab (*In memoriam*) que muito ajudou com sua experiência e prestou ajuda sempre que necessário.
Aos colegas dessa jornada que sempre estiveram presentes e apoiando.
Aos amigos do LESERC (Laboratório de Engenharia de Software e Redes de Computadores) Amanda Serra, Larissa Madeira, Matheus Ferreira, Weslley Lima, Gilberto Nunes,Thiago Pinheiro, Pablo Matos, Gerson Lobato, Osvaldo Junior que se fizeram amigos e me apoiaram e incentivaram.
Aos amigos do LABSAC, Luan Oliveira, Breno Fabrício e Higo Pinheiro.
Ao amigo de trabalho e de vida Steve Ataky, que muito me auxiliou com sua sabedoria nos momentos que precisei, dando força e coragem para enfrentar os desafios propostos nesta etapa de minha vida.
Ao meu amigo Thiago Coelho Ferreira, que durante os últimos meses da realização desta pesquisa apresentou-se como um apoio de total relevância, em critérios pessoais e profissionais, para o término deste trabalho.
Aos professores Dr. Leonardo Paucar, Dr. Nilson Costa, Dr. Allan Kardec, Dr. Francisco Silva, que sempre contribuíram com suas conversas motivacionais para dar continuidade com o trabalho aqui realizado.
Aos amigos de vida Thaynara Cordeiro, Lucimeire Souza pela compreensão do tempo não dedicado a amizade durante a dissertação.
Aos amigos de trabalho que muito contribuíram em minha jornada profissional Marco Monteiro, Arlley Carlos, Raul Garcia, Ullysses Val, Emanuela Vasconcelos, Izidio

Chagas, Thiago Cavalcante, Eduardo Bessa, Eduardo Newton, Otavio Moura, Osvaldo Ferreira, Lennon Joseph.
Aos amigos e professores da UNDB que me acolheram Arikleyton Ferreira, Pedro Brandão, Pedro Henrique, Fábio Feitosa, Alessandro Miranda, Edileide Lima e, em especial, ao coordenador de curso de Sistemas de Informação da UNDB Bruno Lima.
A Universidade Federal do Maranhão (UFMA) e ao Programa de Pós-Graduação em Engenharia de Eletricidade (PPGEE), ao Governo Federal Brasileiro pela oportunidade dada de estudar, oportunidade esta que agregou de forma primordial a minha vida profissional.
À Instituição de fomento CAPES, que auxiliou com a bolsa durante os dois anos percorridos.
A todos que de alguma forma contribuíram direta ou indiretamente para a realização deste trabalho.

"O maior erro é não arriscar por medo de perder
o que ainda nem conquistou".
Autor desconhecido

Lista de Figuras

Lista de Siglas

ATL *Atlas Transformation Language*

EMF *Eclipse Modeling Framework*

MDA *Model-Driven Architecture*

MDE *Model-Driven Engineering*

MOF *Meta Object Facility*

MT4MDE *Mapping Tool for MDE*

PDM *Platform Description Model*

PIM *Platform Independent Model*

PSM *Platform Specific Model*

UML *Unified Modeling Language*

WM *Weaving Model*

WMM *Weaving Metamodel*

XMI *XML Metadata Interchange*

XML *eXtensible Markup Language*

SaaS *Software as a Service*

SO *Sistemas Operacional*

GPS *Global Positioning System*

Conteúdo

1 INTRODUÇÃO

Este capítulo apresenta o contexto em que a dissertação foi desenvolvida, neste irá constar a problemática, a solução e os objetivos almejados. Em seguida, a metodologia de pesquisa aplicada para a execução deste trabalho será apresentada.

1.1 Contexto

Atualmente, a sociedade está imersa em sistemas computacionais que visam auxiliar e trazer comodidade à vida das pessoas. Assim, à medida que novas tecnologias são desenvolvidas, tais como rede sem fio e a computação ubíqua, permitem que a tecnologia facilite e apoie tanto as atividades profissionais quanto o cotidiano das pessoas.

Muitos desenvolvedores vêm desenvolvendo *softwares* com a capacidade de se adaptarem às necessidades da sociedade, visando levar a estes usuários comodidade e comunicabilidade por meio destes sistemas que se adequem ao contexto do usuário. Segundo SATO [9], embora essas tecnologias ofereçam possibilidades para o desenvolvimento de novas categorias de produtos e sistemas, entender os contextos de uso torna-se cada vez mais importante para prever como novos produtos com essa capacidade podem ser aceitos, compreendidos, incorporados, usados e valorizados na vida das pessoas.

As novas tecnologias podem penetrar em nosso espaço de atividades e fornecer serviços funcionais e informativos onde e quando forem necessários. A fim de usá-los efetivamente para melhorar a qualidade da experiência das pessoas, a natureza dos serviços prestados pelo sistema precisa ser cuidadosamente ajustada às necessidades dos usuários e ao contexto de uso [9].

Os sistemas de *software* que se adaptam ao contexto do usuário são sistemas criados com base em *context- aware* e são definidos de *context-aware systems* [22] [23]. Um dos objetivos dos *context aware systems* é adquirir e utilizar informações sobre o contexto de um sistema, a fim de fornecer serviços adequados às pessoas, proporcionar acesso à informação, à comunicação de forma fácil e rápida ao usuário.[10].

Entretanto, o desenvolvimento destes sistemas de *software* possui um alto grau de complexidade. Segundo SATO [8], alguns contextos são compostos de fatores que residem nos usuários, tais como fatores culturais, sociais, cronológicos e cognitivos. Estes contextos são compostos por fatores externos que residem em ambientes operacionais ou organizacionais. Alguns destes contextos também podem ser distribuídos entre diferentes usuários, sistemas ou ambientes operacionais. SATO [8] também diz que :

> " *Diferentes tipos de contextos interagem uns com os outros e compõem camadas subjacentes e complexas de condições operacionais para o desempenho de sistemas interativos que são frequentemente referidos como situações*"(SATO, 2003) [8].

Esta complexidade pode ser entendida a partir do momento que se trabalha com sistemas sensíveis ao contexto, sendo necessário trabalhar com multi plataformas. Plataformas estas que proporcionará seja a distribuição da informação, seja a segurança desta. Afim de entender o sistemas sensíveis ao contexto, VIEIRA [81], faz uma breve comparação entre sistemas tradicionais e sistemas que se adequam ao contexto do usuário.

Os sistemas que não fazem uso de *context aware* são sistemas que atendem a necessidade do usuário, visando unicamente as informações fornecidas explicitamente pelo usuário sem nenhum tratamento específico, todavia, os sistemas que fazem uso de *context aware* entendem as informações fornecidas e são tratadas em uma base de conhecimentos contextuais. Este tratamento leva em consideração as informações cedidas pelo usuário e os dados coletados no ambiente ao qual o usuário está inserido, esses dados são parâmetros que serão usados para tomadas de decisão do sistema [81]. Segundo GRIEBE [3], os sistemas "podem usar esses parâmetros, também chamados de contexto, como vetores de entrada adicionais e ajustar o comportamento da aplicação de acordo com o estabelecido. A utilização destes vetores de entrada adicionais introduz novas fontes de comportamento para esta aplicação". Assim, seguindo a premissa dos autores já citados, se constata que ao desenvolver sistemas que façam uso de *contex-aware* precisa-se entender que estes contextos são adquiridos de fontes heterogêneas; as mudanças no contexto devem detectadas e tratadas; e existe um alto grau de complexidade no desenvolvimento destes sistemas.

Seguindo a literatura [14, 15, 42, 41, 47, 76, 77, 78, 75], vários *frameworks* e *middlewares* foram criados para suportar o desenvolvimento de sistemas sensíveis ao contexto.

Além de *frameworks e middlewares*, algumas abordagens também são citadas na literatura como auxilio à ferramentas que dão suporte ao desenvolvimento de sistemas sensíveis ao contexto, entre essas abordagens pode-se citar a abordagem MDE. Esta abordagem tem como objetivo auxiliar no desenvolvimento de sistemas por intermédio de modelos, possibilitado um nível mais alto de abstração para o desenvolvedores [15].

Segundo SERRAL et al[15], da mesma forma que o primeiro compilador FORTRAN foi um grande marco na ciência da computação, pois pela primeira vez esta linguagem de programação deixou que os programadores especificassem o que a máquina deveria fazer, a MDE [15]:

> " *é um passo além, no qual o software é desenvolvido não diretamente escrevendo código em linguagens de implementação,mas, especificando o sistemas utilizando modelos em um nível mais alto de abstração, que pode ser transformado em código por gerações de código automatizadas*"(SERRAL, 2009) [15].

Acredita-se que a utilização de *frameworks* ou *middlewares* baseados em uma abordagem dirigida por modelos possam auxiliar consideravelmente o desenvolvimento de sistemas sensíveis ao contexto. Tal ação minimizaria os possíveis problemas como a complexidade de se trabalhar com multi plataformas e o desenvolvimento de *software* que é suportado por abstrações, i.e. modelos.

1.2 Problemática

Segundo BIEGEL [47], os *context aware system* são um grande e importante subconjunto do conjunto geral de aplicações de computação ubíqua, e já demonstraram as vantagens obtidas com a capacidade de perceber o ambiente circundante [4] [5] [6]. No entanto, estes sistemas ainda permanecem difíceis de desenvolver e implementar, o que por sua vez exige dos programadores escrever grandes quantidades de código e interagir com dispositivos de sensores e atuadores em um nível baixo de programação, a fim de desenvolver aplicações relativamente simples [47].

Quando nos referimos sobre a complexidade no desenvolvimento de *context aware system*, estamos abordando um amplo e abrangente problema que pode ser retratado e tratado de diferentes formas. Pode-se trabalhar a complexidade no desenvolvimento a partir do ponto de vista de sistemas distribuídos, escalonamento, algoritmos complexos, plataformas heterogêneas, etc.

Um dos problemas considerando *context aware system* é a necessidade de trabalhar com plataformas heterogêneas. Ao se trabalhar com essas plataformas, diversas variáveis, dados e informações são coletadas, a partir da coleta, essas informações são processadas e realizada a tomada de decisão de acordo com o algoritmo específico. Assim, este trabalho de pesquisa aborda o problema de complexidade no desenvolvimento de sistemas sensíveis ao contexto para plataformas heterogêneas.

1.3 Motivação

A possibilidade de trabalhar com a abordagem MDE que utiliza um nível alto de abstração, flexibilidade e possibilidade de reengenharia seria propícia para os programadores que necessitam desenvolver sistemas que utilizam *context aware*.

A partir dessa solução, muitas pesquisas tais como vista nos trabalhos de [77], [78], [76], foram realizadas com o objetivo de propor algo que auxilie no desenvolvimento de *context aware systems*. Entre as soluções propostas nos trabalhos disponíveis na literatura, está incluso a MDE. Porém, o que é possível perceber diante do exporto na literatura [8] [9] [15] [47] [81] [4], é que a complexidade no desenvolvimento de *context aware systems* ainda é pertinente, principalmente quando se trabalha com plataformas heterogêneas.

A MDE poderá trabalhar em um nível de abstração mais alto, possibilitando também maior gerenciamento dos problemas de produtividade, portabilidade, interoperabilidade presentes no desenvolvimento destes sistemas, onde as plataformas heterogêneas serão retratadas por meio de modelos. Esses modelos serão gerenciados através da MDA para suportar o desenvolvimento de sistemas de *softwares*. Como as plataformas

heterogêneas serão representadas por modelos, existe a necessidade que se faça um entrelaçamento entre os elementos do modelo contendo a lógica do negócio (i.e. PIM) com os elementos do modelo contendo os aspectos próprios de uma plataforma (i.e. PDM) para que se possa gerar o modelo específico de uma plataforma (i.e. PSM).

Alguns trabalhos disponíveis na literatura, [65] [64], já utilizam a técnica de *Weaving* para entrelaçar modelos heterogêneos, porém estes são entrelaçados de forma serial. Nosso trabalho propõe um *framework* capaz de fazer o entrelaçamento paralelo de modelos, que auxilia no desenvolvimento de sistemas sensíveis ao contexto baseado em MDE e *Weaving*.

1.4 Objetivos

1.4.1 Objetivo Geral

O objetivo geral da pesquisa é fornecer um *framework* baseado em MDE e *Weaving* para suportar o desenvolvimento de sistemas sensíveis ao contexto.

1.4.2 Objetivos Específicos

Os objetivos específicos são:

- propor um *framework* baseado em MDE e técnica de *Weaving* para suportar o desenvolvimento de sistemas de *software* sensíveis ao contexto;
- aplicar o *framework* para integração de plataformas heterogêneas para o desenvolvimento de aplicações sensíveis ao contexto;
- aplicar a técnica de *weaving* para fazer o entrelaçamento paralelo de modelos;
- definir transformações a partir do entrelaçamento dos modelos utilizados;
- fornecer um exemplo ilustrativo para validar a proposta do *framework*.

1.5 Metodologia de Pesquisa

A metodologia utilizada para esta pesquisa é apresentada como a seguir::

1. Coleta de informações através de pesquisa bibliográfica realizada por meio de sites como :*IEEE, ACM, Wiley, SpringerVerlag* sobre os referidos temas:

   - Engenharia Dirigida por Modelos e suas respectivas definições, linguagem de transformação, metamodelos e modelo, [38], [30], [43];
   - técnica de *Weaving* e seu funcionamento, [65], [64];
   - ciência de contexto e suas principais funcionalidade e definições de uso, [17], [28] ;
   - ATL e suas definições e funcionamento, [39].

2. Levantamento de trabalhos relacionados à pesquisa disponíveis na literatura, onde estes foram relacionados de acordo com o foco da pesquisa e selecionados como se segue:

   - trabalhos relacionados à abordagens baseadas em MDE para o desenvolvimento de sistemas de *software*, [42] [41] ;
   - buscou-se trabalhos relacionados à abordagens para o desenvolvimento de sistemas sensíveis ao contexto, [47] [76] [77];
   - trabalhos que condizem com MDE e Sistemas sensíveis ao contexto, [75] [78].

3. Proposta de um *framework* baseado em MDE e *Weaving* para suportar o desenvolvimento de sistemas sensíveis ao contexto, , onde nesta proposta se utilizou:

   - *Eclipse Modeling Framework Project* (EMF) para a construção de modelos e metamodelos necessários para a implementação do *Framework*;
   - construção de PDMs com base nos metamodelos de Contexto, Segurança e Sistemas Distribuídos;
   - construção do PIM, que será a regra de negócio;
   - construção da ferramenta com base em *weaving* que possa entrelaçar os modelos, ferramenta esta desenvolvida em linguagem Java.

4. Análise dos trabalhos relacionados [42, 41, 47, 76, 77, 78, 75] de acordo com os itens abaixo:

    - tipo de abordagem utilizada no desenvolvimento de *softwares*;
    - mecanismo utilizado para unir as abstrações do sistema de *software*;
    - processo de desenvolvimento para aplicações sensíveis ao contexto;
    - permitir correspondências entre elementos de bases/plataformas heterogêneas;
    - geração automática ou semi-automática de código fonte;
    - reutilização de modelos;
    - capacidade de trabalhar com plataformas heterogêneas em paralelo;
    - suporte ao desenvolvimento de sistemas de *software*.

5. Construção de um protótipo de ferramenta que implementa o *framework* proposto para mostrar a viabilidade do mesmo.

1.6 Apresentação da Dissertação

Este trabalho de dissertação de mestrado é apresentado como a seguir.

O primeiro Capítulo aborda o contexto no qual está inserido o tema da Dissertação, a problemática da dissertação, as tecnologias e abordagens presentes na solução. Apresentou-se o objetivo geral, os objetivos específicos, a motivação e a metodologia de pesquisa.

O segundo Capítulo fala-se sobre a Fundamentação Teórica, descrita a partir dos conceitas e notações dos principais temas que fazem parte desta pesquisa. Apresenta-se a contextualização da Abordagem *Model Driven Engineering*,*Archtecture Driven Engineering*, *Eclipse Modeling Framework* Modelo *Weaving*,*Atlas Transformation Language* Ciência de Contexto.

O terceiro Capítulo o estado da arte é apresentado, bem como, os trabalho relacionados que focam no processo de desenvolvimento baseado em MDE de sistemas de *software*, assim como, as comparações realizadas entre esses trabalhos e trabalho presente nesta Dissertação.

O quarto Capítulo apresenta a o *framework* proposto, bem como, seus metamodelos para suportar a integração das aplicações utilizadas pelo *framework*. Também são apresentadas a forma que foram integradas e as definições de transformações para a criação dos modelos que servirão de suporte ao demais modelos mais refinados e posterior ao código fonte.

O quinto Capítulo apresenta a arquitetura do *framework*, a implementação e o entrelaçamento realizado através da ferramenta construída para integrar os modelos heterogêneos.

O sexto Capítulo mostra um exemplo ilustrativo que auxilia na compreensão do funcionamento da ferramenta que implementa o *framework* proposto.

Finalizando a dissertação, o sétimo Capítulo apresenta as conclusões da pesquisa, suas devidas contribuições e limitações, principais dificuldades encontradas e os trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta a fundamentação teórica serve de base para a concepção do *framework* e criação do protótipo da ferramenta que suporta o desenvolvimento de sistemas sensíveis ao contexto.

Ainda neste capítulo, abordaremos os conceitos de MDE *(Model Driven Engineering)*, assim como, os conceitos de MDA *(Model Driven Architecture)*, EMF *(Eclipse Modeling Framework)*, *Weaving* e Ciência de Contexto.

2.1 Model-Driven Engineering (MDE)

A MDE surgiu como uma abordagem para enfrentar alguns dos principais problemas do desenvolvimento de softwares, como: a complexidade das plataformas e as limitações de linguagens como Java ou C# [19]. A MDE é uma abordagem de desenvolvimento de software que se concentra na criação de modelos que descrevem os elementos de um sistema [19] e orientam sua implementação [25]. Entre as vantagens encontradas nesta abordagem, pode-se citar a produtividade, a manutenibilidade e a interoperabilidade [24].

Segundo LEFI [70], a pesquisa e a prática da Engenharia Dirigida por Modelos (MDE) "progrediram significativamente na última década para lidar com o aumento da complexidade dentro dos sistemas durante seus processos de desenvolvimento e manutenção."A MDE pretende elevar o nível de abstração usando modelos como entidades de primeira classe e, consequentemente, os artefatos primários do desenvolvimento [70].

No centro da abordagem MDE está a transformação de modelos que permite definir como um conjunto de elementos de um metamodelo de origem é analisado e transformado em um conjunto de elementos de um metamodelo-alvo [70].

Modelo é uma descrição de um objeto, coisa, a qual deve ser seguida em um projeto. Segundo KLEPPE [30], modelos respeitam um formato preciso (uma sintaxe) e um

significado (uma semântica), tal descrição deverá ser útil para a interpretação computacional. Para se criar modelos é necessário que haja uma linguagem bem definida que possua um metamodelo. Metamodelo é "um modelo que define a linguagem para exprimir um modelo " [31]. Um metamodelo é definido por um Metametamodelo que é similar à relação entre um modelo e metamodelo[31].

Nesta dissertação, utiliza-se MDE, especificamente alguns conceitos de MDA como PIM, PSM e definição de transformação entre modelos, como base para o *framework* proposto.

2.1.1 Arquitetura Dirigida por Modelos (MDA)

A OMG (*Object Management Group*) definiu MDA como um *framework* de desenvolvimento de *software*. Em MDA, o ciclo de vida de um *software*, isto é, sua concepção, desenvolvimento, manutenção e evolução é baseado em modelos formais que são entendidos e manipulados pelo computador [30].

No contexto de MDA, alguns conceitos são importantes tais como[35]:

- **Modelo Independente de Plataforma (PIM) :** Modelo de alto nível de abstração, independente de aspectos tecnológicos de implementação da solução;
- **Modelos Específico de Plataforma (PSM):** segundo CARVALHO [35] o PSM é obtido a prir das definições de transformações do modelo PIM. O PSM deverá descrever os aspectos tecnológicos da implementação;
- **Definição de Transformação:** é um conjunto de regras de transformação que descrevem como transformar os elementos do modelo fontes nos elementos do modelo alvo;
- **Motor de Transformação:** faz a execução das definições de transformação de modelos tendo como entada um modelo e gerando como saída um outro modelo;

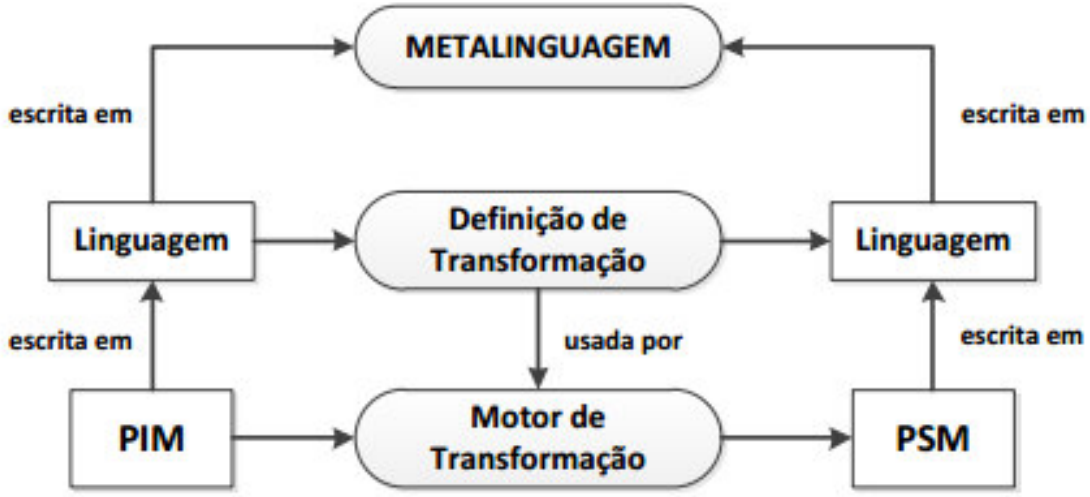

Figura 1: Framework básico da MDA [30]

Segundo KLEPPE [30], a MDA fornece o gerenciamento dos problemas de produtividade, portabilidade, interoperabilidade presente no desenvolvimento de *software* conforme descrito a seguir:

- **Produtividade:** em MDA, os desenvolvedores centram-se na criação do PIM (Modelo Independente de Plataforma), e o PSM (Modelos Específico de Plataforma) adentrado na definição de transformação. Para mais, a maior parte do código é gerado a partir da transformação do PIM para o PSM, deixando assim, o mínimo de código a ser escrito no nível do PSM;

- **Portabilidade:** o fato de um mesmo PIM poder se transformar em diferentes PSMs através das definições de transformações, resultará em PSMs de plataformas heterogêneas,permitindo a Portabilidade;

- **Interoperabilidade:** a interoperabilidade é alcançada por meio de criações de elos entre o PSM gerado a partir de um mesmo PIM. Estes elos permitirá que conceitos de uma plataforma seja transformado em conceitos de outra plataformas [35].

Na Figura **??**, podemos observar uma transformação de acordo com a abordagem MDA. Mediante sucessivas transformações que durante o processo, como resultado final

é obtido o código-fonte do processo de desenvolvimento.

Figura 2: Processo de Transformação da MDA [30]

A principal proposta da MDA é garantir a gestão do ciclo de vida do desenvolvimento de *software*. Desta forma, têm-se um alto nível de abstração, manutenção e evolução do *software* na criação e manutenção de modelos, [31].

A Figura 3 apresenta os níveis de modelos utilizados por MDA e EMF.

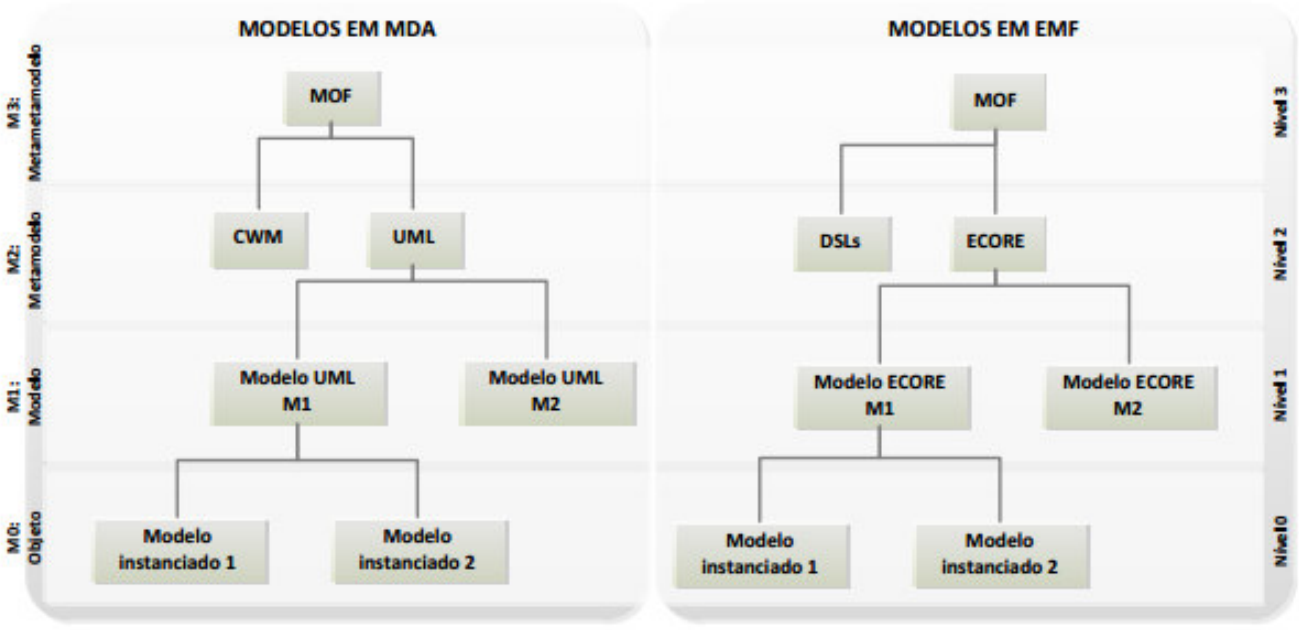

Figura 3: Descrição dos Níveis de camadas de modelagem - [88] [31]

Segundo KLEPPE [30] na Figura 3 é descrito os níveis de camadas de modelagem, contendo:

- M0 (objeto): representa as instâncias do conceito do mundo real. O principal objetivo é descrever objetos de um domínio computacional;
- M1 (modelo): camada dos modelos do mundo real, representados conforme definido no metamodelo correspondente. Objetivo principal é definir o domínio da informação por meio de uma linguagem que descreva este modelo;
- M2 (metamodelo): camada dos metamodelos, seu principal objetivo é definir a linguagem que especifique os modelos;
- M3 (metametamodelo): camada de alto nível, constitui a arquitetura de metamodelagem. Nesta defini-se uma linguagem abstrata e um *framework* para representar metamodelos.

2.1.2 Eclipse Modeling Framework (EFM)

Segundo STEINGERG [29], EMF é um *framework* de modelagem que implementa conceitos da abordagem MDE para a ferramenta Eclipse. Neste trabalho utilizamos o EMF para a criação dos modelos e metamodelos e na transformação de modelos.

De acordo com [50], o EMF tem como propósito unir o uso das tecnologias como Java, *extensible Markup Language (XML)* [84] e *Unified Modeling Language (UML)*[52] Figura 4.

Também pode-se considerar o EMF como um *framework* facilitador de modelagem e geração de código para construção de ferramentas ou aplicações, onde para isto possui em a estrutura de um modelo de dados como sua base [88]. Um modelo em EMF possui diversas formatos, tais como: Ecore, que é um subconjunto do metamodelo UML [29], possui o formato XMI que é o padrão da OMG [55] por onde troca informações baseado em XML e através de Java utilizando *@annotations.*

2.1.3 Especificação de Correspondência e Definição de Transformação

As transformações de modelos se dividem em duas etapas, de acordo com LOPES [39] elas são : a especificação de correspondência e da definição de transformação;

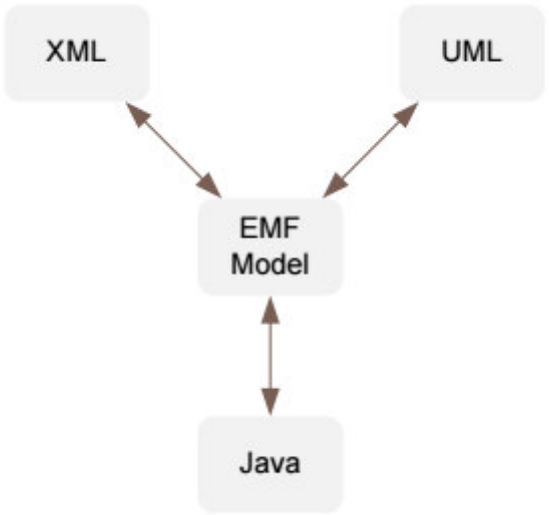

Figura 4: EMF e suas tecnologias [53]

- a especificação de correspondência: responsável por mapear os relacionamentos entre os metamodelos;
- definição de transformação: após o mapeamento realizado na especificação de correspondência, regras de transformação são descritas para que haja a transformação de um modelo fonte para o modelo alvo.

LOPES [39] mostra um modelo proposto de uma arquitetura para transformações de modelos, arquitetura a qual se diferencia visivelmente os modelos de correspondência e transformação, esta pode ser vista na Figura 5.

A arquitetura pode ser explicada como se segue [39][69]:

- **MMM:** corresponde ao metamodelo como exemplo, Ecore;
- **MMfonte e MMalvo:** refere-se aos metamodelos do qual está partindo a transformação para o qual será transformado;
- **M do fonte e M do alvo:** aqui nos referimos as camadas de modelos, por exemplo: uma aplicação que precisa obter dados de contexto, como: bateria, memória e cpu que posterior será transformado em um modelo com as correspondência dos modelos alvos como: modelos de sistemas distribuídos e segurança;

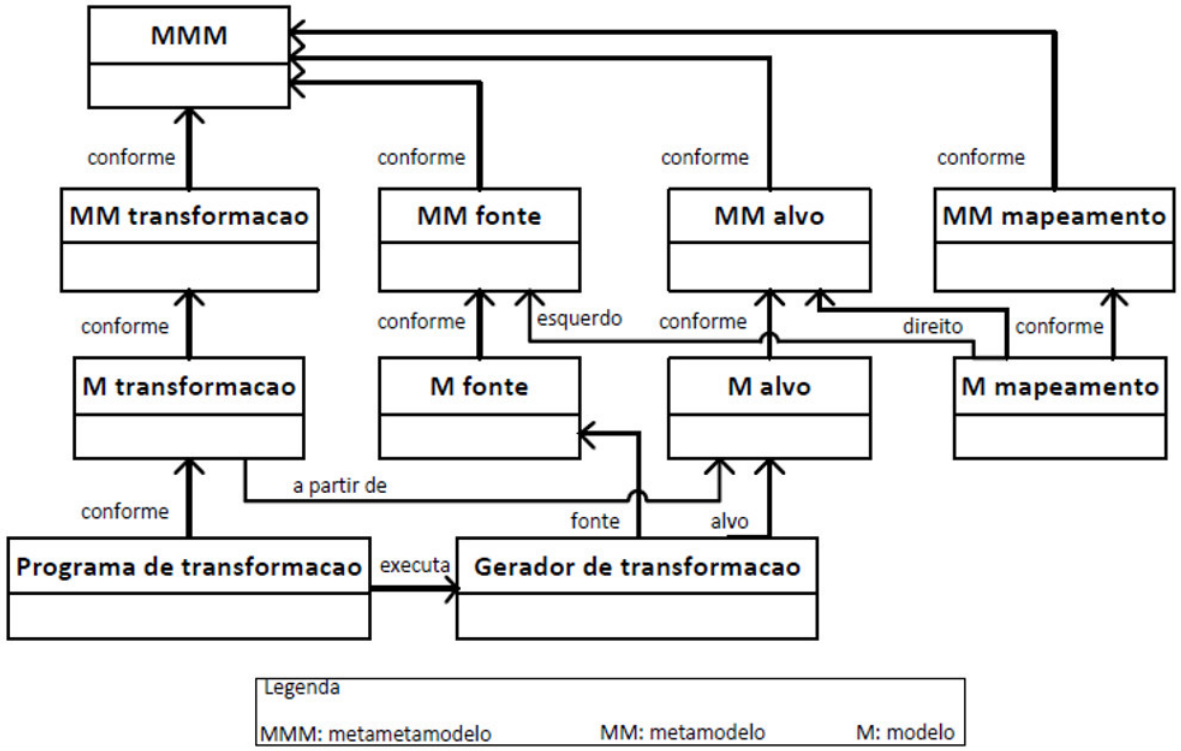

Figura 5: Arquitetura de transformação de modelos[39]

- **MM mapeamento:** metamodelagem de correspondência, utilizando as correspondências entre os elementos de um metamodelo fonte e os elementos de um metamodelos alvo [69];

- **M mapeamento:** modelo que armazena as correspondências entre dois metamodelos [69];

- **MM transformação:**baseia-se em "uma linguagem de transformação que permite a criação exata de transformações de modelo fonte ao modelo alvo"[69];

- **M transformação:** modelos de transformação que demonstra como os elementos de um metamodelo fonte são transformação em metamodelo alvo;

- **Programa de transformação:** uma ferramenta que realiza as transformações necessárias de um modelo fonte ao modelo alvo.

Sabe-se que existem muitas linguagem de transformação que podem ser utilizadas dentro do contexto de MDE, dentre elas podemos citar : MOF QVT, MOFScript e a

ATL, esta última foi a escolhida a ser utilizada nesta dissertação.

De acordo com [74], *ATL - Atlas Transformation Language* "é uma linguagem de modelo de transformação e *toolkit*. No campo da Engenharia Dirigida por Modelos (MDE), ATL fornece maneiras de produzir um conjunto de modelos alvo de um conjunto de modelos de origem. ". Neste trabalho, o ATL foi usado no processo de definição de transformação dos diferentes modelos utilizados no quadro proposto . Os três mudanças de configurações são criados para cada processo. Cada processo define uma regra de transformação, a primeira refere-se ao modelo de transformação, o texto e, o segundo modelo ao código.

2.2 Modelo de Weaving

De acordo com FABRO [7], *Weaving* é "uma operação genérica que estabelece correspondências com significado semântico entre elementos de modelo complexos". A realização desta transformação é feita por uma ferramenta de transformação baseada em linguagem de transformação onde neste trabalho utilizamos da ATL *(ATLAS Transformation Language)* [66].

Segundo [64] na tecelagem de modelos o próprio usuário poderá definir "a semântica do meta-modelo de tecelagem através da criação de tipos que serão associados às ligações entre elementos de modelos tecidos". Ainda segundo [65] as "transformações de modelos podem ser utilizadas para resolver alguns problemas através da especificação de traduções automáticas de uma representação para outra. A criação de um programa de transformação, entretanto, não é automática e tem um certo custo". A Figura 6 apresenta um contexto da tecelagem de modelo[65].

A tecelagem tem como objetivo formular ligações entre metamodelos, estes por sua vez são representações de elementos MMEsquerda e MMDireita. o MMEsquerda representa o metamodelo fonte e o MMDireita representa o metamodelo alvo. As ligações estabelecidas entre esses elementos geram o modelo de tecelagem (WM). O modelo WM deve ser conforme o metamodelo de tecelagem (WMM) especificado [64]. Sabe-se que

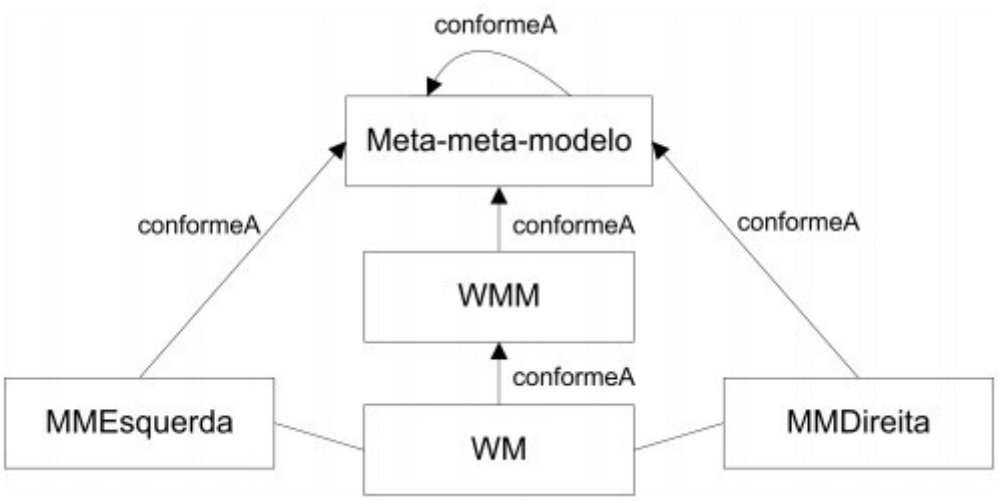

Figura 6: Contexto de tecelagem de modelos [65]

não existe um metamodelo de tecelagem padrão que atenda a todas as necessidades, entretanto [65] et al afirmam que existem similaridades entre os diversos metamodelos de tecelagem existentes, permitindo assim que seja criado um núcleo genérico de metamodelo de tecelagem. Ele propõem um metamodelo base que contém os elementos necessários para a tecelagem de modelos 7, metamodelo este que pode ser estendível e que serviu de base para o metamodelo *weaving* presente neste trabalho. [65].

Segundo [65] o metamodelo pode ser explicado da seguinte forma:

- **WElement:** elementos base de todos os demais elementos presentes no metamodelo. Este elemento tem dois atributos: nome (name) e descrição (description);
- **WModel:** elemento raiz do metamodelo de tecelagem. Composto de elementos de tecelagem e referências para modelos tecidos;
- **WLink:** representa a ligação entre elementos de modelos. Faz referência *end* e possibilita fazer ligações com um número arbitrário de elementos. Este pode ser estendido para que possa adicionar diferentes ligações semânticas ao metamodelo.
- **WLinkEnd:** indica a extremidade de uma ligação fazendo a referência de modelo tecido por meio do *WElementRef*;

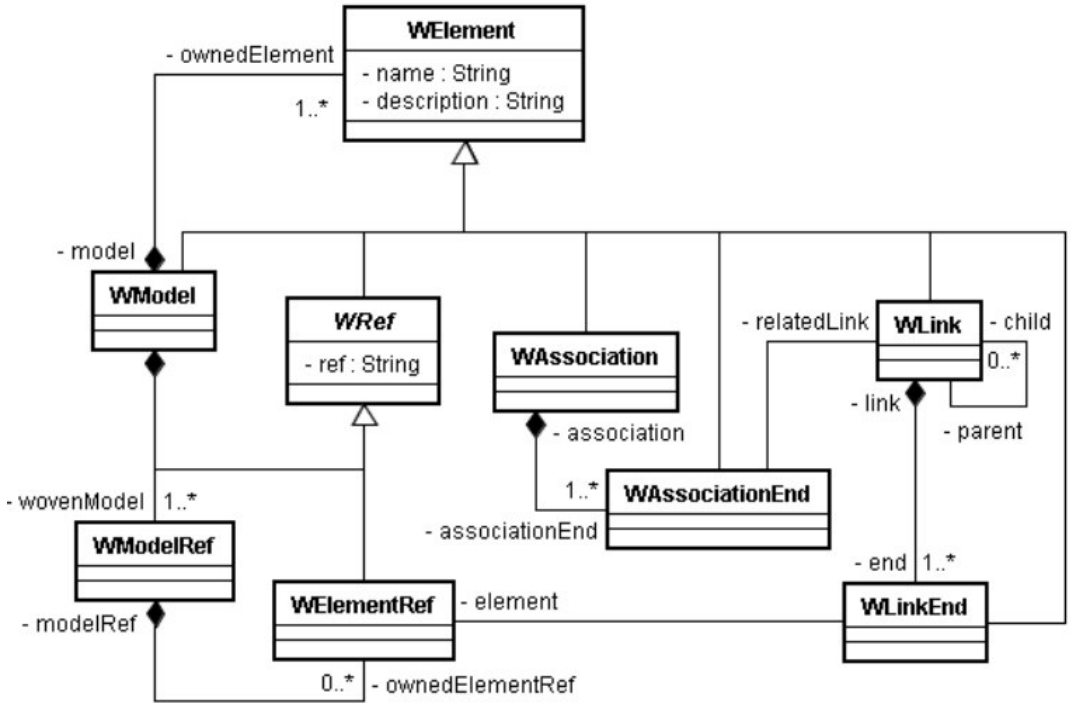

Figura 7: Metamodelo de tecelagem genérico [65]

- **WElementRef:** é o elemento que é referenciado de um modelo de tecelagem. O atributo*ref* dispõe do identificador do elemento tecido;
- **WAssociation:**cria relacionamentos de associação entre as ligações;
- **WAssociationEnd:** parecido com o WLinkEnd, porém este especifica as extremidades de uma associação.

2.3 Ciência de Contexto

Uma das primeiras discussões a respeito de computação ciente de contexto defende o conceito de ciência de contexto é "a capacidade de aplicativos móveis descobrir e reagir às mudanças no ambiente em que estão situados "[13]. Segundo o Dicionário [33], contexto é definido como “ condições inter-relacionadas nas quais alguma coisa existe ou ocorre ”. O contexto é o que fundamenta a habilidade de identificar o que é e o que não é relevante em um dado momento [32]. Seguindo estas definições entende-se que contexto é a obtenção e tratamento do estado ao qual o usuário esteja imerso, ou seja, são as informações que pertinentes ao ambiente o qual o usuário está inserido.

Segundo BRÉZILLON [26], contexto é um conjunto de condições relevantes e influências que fornecem o entendimento de uma determinada situação, onde estas condições e influencias estão diretamente sobre as entidade do domínio. Segundo DEY [28] esta entidade pode ser uma pessoa, um lugar, ou um objeto que é considerado relevante para a interação do usuário com a aplicação.

Outros pesquisadores definem contexto por meio de categorização de tipos de informações relacionadas ao ambiente, como por exemplo [48], que define duas categorias de contexto : *interno* que seriam as informações provenientes no estadio o qual o usuário se encontra e o *externo* que são as informações sobre o ambiente o qual o usuário está sendo inserido ou já está inserido.

Entretanto, pode-se dizer que a definição mais aceita pelos pesquisadores é a de [27], que se refere à Ciência de contexto como:

> " Contexto é qualquer informação que pode ser utilizada para caracterizar a situação de uma entidade. Uma entidade é uma pessoa, lugar, ou objeto que é considerado relevante para a interação entre um usuário e uma aplicação, incluindo o usuário e a aplicação em si" DEY [27].

Contexto envolve informações que se referem a vários aspectos associados ao funcionamento da aplicação, tais como o usuário, o dispositivo de acesso, o ambiente, a rede, dentre outros [28]. Com estas definições se percebe o quanto Contexto está presente no dia a dia das pessoas, seja em seus carros através do GPS *(Global Positioning System*), sensores que adequam o ar condicionado do carro de acordo com a temperatura do corpo do motorista ou mais comumente percebido em *smartphones* através dos sistemas de economia de energia, aplicativos de localização, ou mesmo plataformas desenvolvidas para suportar aplicações sensíveis a contexto.

No trabalho de CHEN [34], quatro categorias de contexto podem ser descritas como apresentado na Figura 8.

Segundo VIEIRA et al [56], o desafio da ciência de contexto é a interação explícita de usuários como o sistema para obter o que se deseja, onde a ciência de contexto

Contexto	Descrição
Contexto computacional	Conectividade de rede, largura de banda, custo computacional e recursos próximos como impressoras e terminais.
Contexto do usuário	Preferências do usuário, localização, mobilidade do usuário, pessoas próximas.
Contexto físico	Temperatura, luminosidade, pressão e umidade.
Contexto de tempo	Dia, mês ou hora.

Figura 8: Categoria de Contexto [34]

visa diminuir esta interação. Sabendo-se do desafio que o a Ciência de Contexto tenta solucionar, muitas pesquisas são realizadas para que haja uma evolução nesta área obtendo o máximo de resultados pródigos possíveis.

Toda aplicação sensível ao contexto necessita obedecer funcionalidades básicas para que seja realizada sua construção. Estas funcionalidades são listadas por [57], onde estão definidas em 3 categorias:

- **Especificação do Contexto:** parte de levantamento de requisitos de contexto e da modelagem das informações contextuais que se faz necessários para o desenvolvimento da aplicação;
- **Gerenciamento de Contexto:** informa como o contexto deve ser manuseado pelo sistema, nos seguintes aspectos: aquisição, processamento, armazenamento e disseminação de Elementos Contextuais;
- **Uso de Contexto:** define a influência do contexto no comportamento do sistema e como este será utilizado.

Estas categorias podem auxiliar no processo de desenvolvimento de plataformas sensíveis ao contexto, categorias estas que serão aplicadas neste trabalho. De acordo com [68][72] para contextualizar uma determinada atividade é necessário verificar um conjunto de dimensões básicas que são:

- **Who (quem):** é importante obter informações contextuais de todas as pessoas que participaram em uma atividade assistida por computador;

- **What (o que):** Identifica o que o usuário está fazendo;
- **Where (onde):**Identifica a localização onde o usuário está inserido. Este é o mais comumente utilizado em sistemas sensíveis ao contexto;
- **When (quando):** indexa um atividade capturada a partir de um contexto temporal ou informa por quanto tempo o usuário permaneceu em um local;
- **Why (porquê):** este é um desafio para a computação móvel, pois é difícil entender o porque um usuário decidiu por determinada ação. Entretanto as informações de contexto coletadas devem inferir os motivos pelos quais o usuário fez determinada ação;
- **How (como):** associa-se ao Why a dimensão How para inferir informações sobre Why.

Além de entendermos os conceitos de Ciência de Contexto, também faz necessário entender suas respectivas habilidades, como no caso da obtenção de informação contextual. Neste sentido a informação pode ser classificada de acordo de como ela é obtida [16].:

- **Sentida:** contexto adquirido por meio de sensores que coletam informações do ambiente tais como : temperatura, presença etc;
- **Derivada:** contexto adquirido em tempo de execução;
- **Provida:** informações de contexto que são fornecida diretamente pelo usuário.

Independente de como a informação de contexto seja obtida, essas informações passam por um processo de obtenção que não é fácil ser construído. Além disso, as informações contextuais são dinâmicas, o que torna as aplicações mais complexas de serem desenvolvidas e por sua vez também difíceis das mesmas gerenciar todos os aspectos de contexto.

2.4 Síntese

O Capítulo teve como foco mostrar os assuntos abordados durante o desenvolvimento desta pesquisa. Temas como MDE, *Weaving* ,ATL, metamodelo, modelos e MDA foram pesquisados e explorados neste Capítulo.

Neste capítulo, uma visão sobre os principais conceitos e tecnologias utilizadas nesta pesquisa foram apresentados. A principal finalidade deste capítulo é expor os temas relacionados com a pesquisa para uma melhor compreensão do *framework* proposto.

Desta forma o Capítulo retrata o estudo bibliográfico realizado sobre as áreas do conhecimento necessários para o desenvolvimento desta dissertação.

3 ESTADO DA ARTE

Este Capítulo tem como foco mostrar os trabalhos que estão disponíveis na literatura para suportar o desenvolvimento de sistemas sensíveis ao contexto.

Também neste Capítulo, serão analisados os trabalhos relacionados a pesquisa desta Dissertação.

3.1 Abordagens baseadas em MDE para o desenvolvimento de sistemas de software

Nesta seção, alguns trabalhos baseados em MDE para suportar o desenvolvimento de sistemas de software são analisados.

3.1.1 A Model Driven Engineering Approach to Support the Development of Secure Software as a Service [42]

MATOS el al[42] propõem um *framework* baseado em MDE para suportar o desenvolvimento de sistemas de *software* para computação em nuvem do tipo SaaS (*Software as a Service*).

Segundo MATOS et al [42] a proposta do *framework* pretende alcançar uma adaptação entre modelos e auxiliar na condução de processos de desenvolvimento de *softwares*. O *framework* proposto neste trabalho faz uso da abordagem MDE aliada a técnicas de entrelaçamento de modelos do domínio de segurança a modelos do domínio da aplicação.

O trabalho de MATOS et al [42] propõem um *framework* baseado em MDE e *Weaving* para entrelaçar os PIMs e PDMs. O entrelaçamento compreende entrelaçar elementos de dois ou mais modelos através de correspondências entre os elementos de seus respectivos metamodelos ou análise de características funcionais dos modelos. As definições de transformação de modelos tomam um PIM, PDM e modelo de *Weaving* como entrada e geram o PSM como saída.

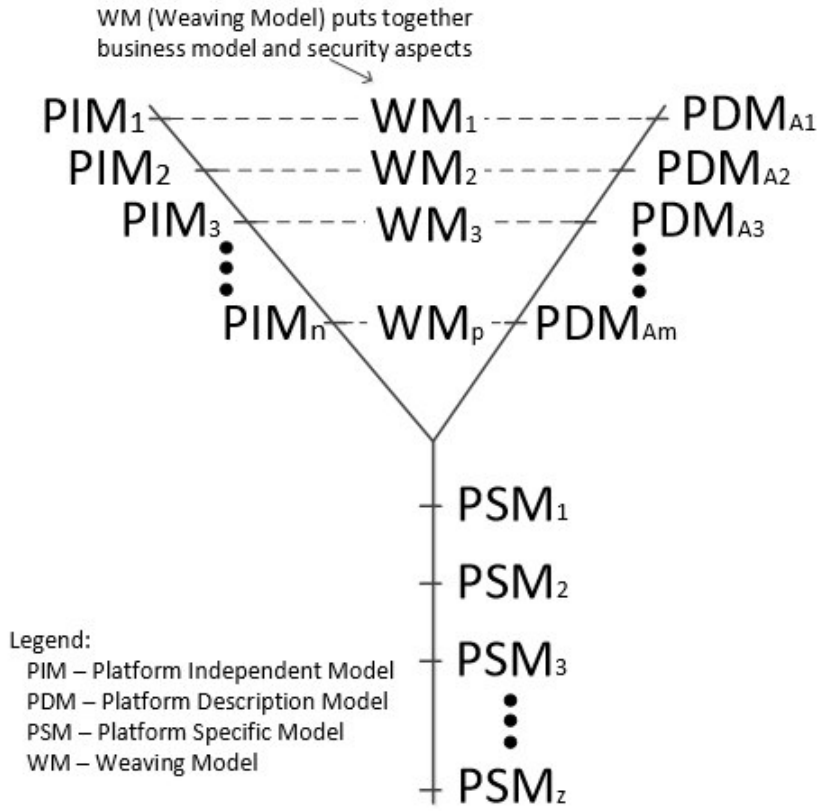

Figura 9: MDE, modelos *weaving*, processo de desenvolvimento em Y [37]

A estrutura utilizada neste processo é baseado em [37], conforme Figura 9.

Em seu trabalho a utilização do Metamodelo *Weaving* (Figura 10) se deu para que pudesse ocorrer o entrelaçamento entre os modelos PIM e PDM, após o entrelaçamento novas transformações são executadas até obter o código.

Ainda seguindo o autor [69] "*através dos recursos fornecidos pela EMF foram desenvolvidos metamodelos com o objetivo de trazer para o ambiente da MDE os sistemas utilizados, as plataformas alvo, e as ferramentas responsáveis por estabelecer os relacionamentos identificados.*"

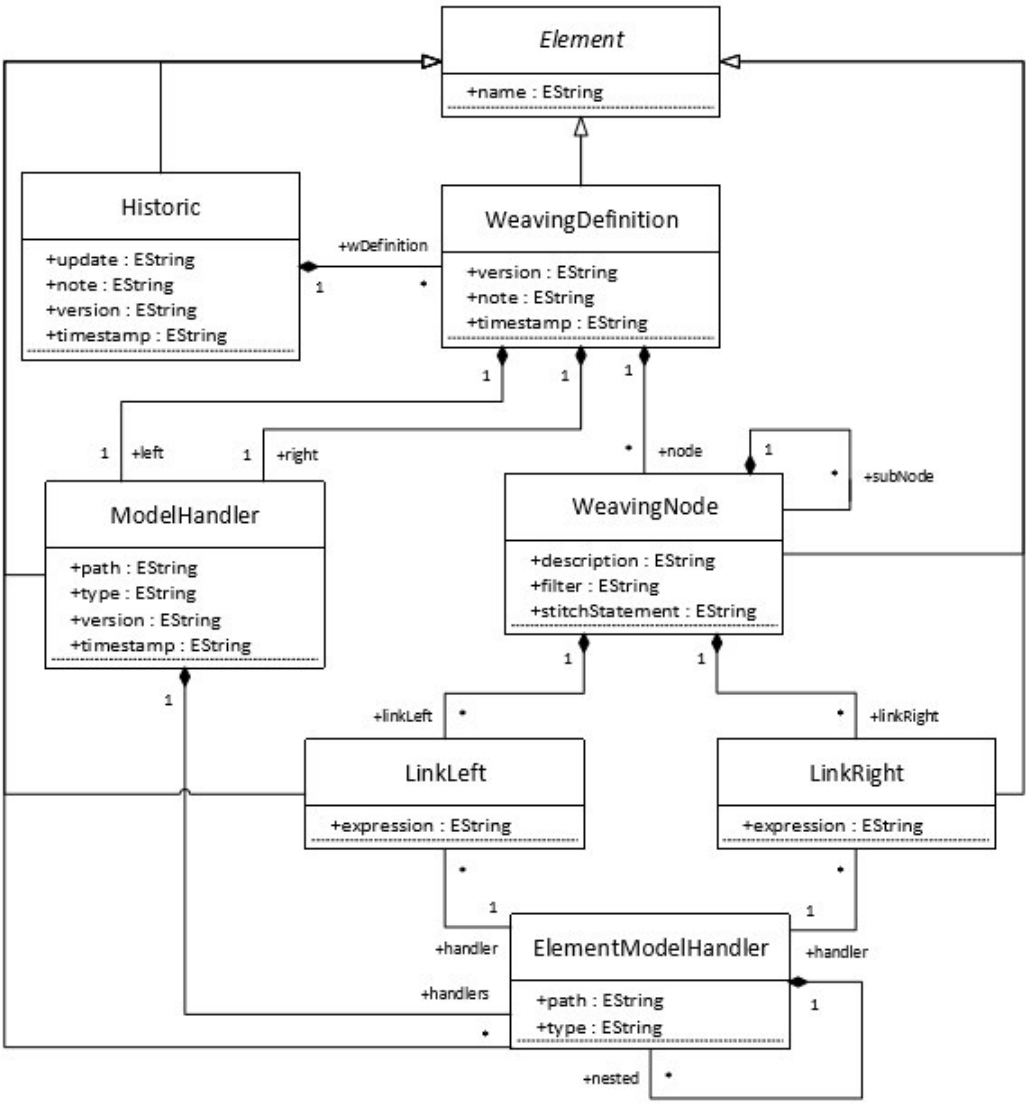

Figura 10: Metamodelo *Weaving* baseado em [42]

3.1.2 A Framework Based on Model Driven Engineering to Support Schema Merging in Database Systems [41]

CARVALHO et al [41] propõem um *framework* baseado em MDE para suportar o processo "de identificação de correspondências entre modelos de base de dados com o intuito de fazer a fusão entre os mesmos".

O artigo tem como principal foco criar uma visão unificada para o problema de identificar elementos correspondentes entre dois esquemas de base de dados. O *framework* desenvolvido pretende auxiliar o especialista de domínio na tarefa de integração de es-

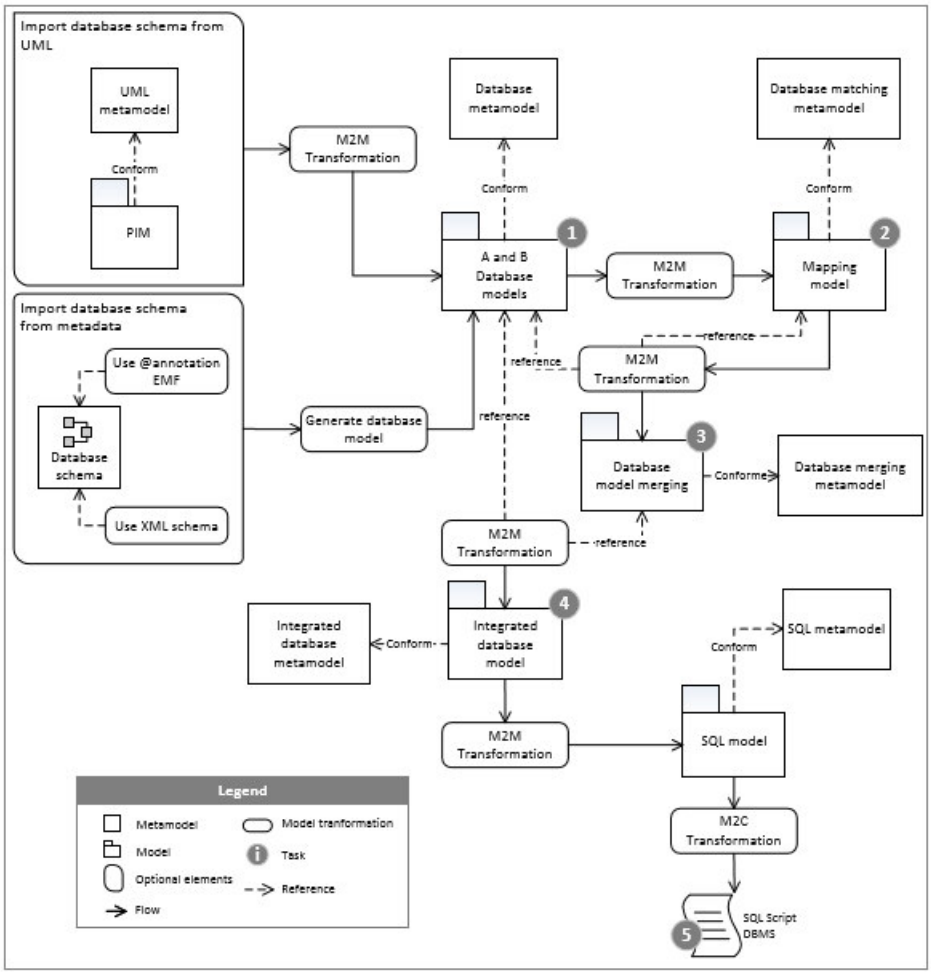

Figura 11: Framewok de suporte a base heterogêneas [41]

quemas de dados de forma semiautomática. O *framework* informa possíveis elementos que são correspondentes aos esquemas de base de dados e o especialista de domínio confirma as correspondências. Na Figura 11 é apresentado o *framework* responsável por tal ação.

Na Figura 11 pode se observar que o produto final que se quer adquirir é o *script* SQL com a própria estrutura do banco de dados que possui a fusão realizada através da fusão (merge) dos modelos de base de dados.

CARVALHO et al [41] descreve o *framework* como se segue:

- **Obter modelos dos esquemas de base de dados:** são modelos construídos a partir do metamodelo de *database*, este representam os modelos de entrados que

serão posteriormente correspondidos;

- **Obtenção de *mapping model*:** modelo conforme metamodelo de *database matching*, sua principal finalidade está nos registros de correspondências;
- **Obter database model merge:** modelo construído a partir do metamodelo *database merging*, sua principal função é registrar os elementos que irão compor o *integrated database model*;
- **Integrated database model:** modelo construído conforme o metamodelo *integrated database*, principal finalidade é a especificação dos elementos que estão como parte do *script* SQL que deverá ser gerado.

No trabalho, o autor utiliza a abordagem MDE para o processo de desenvolvimento e utiliza o *merging* de modelos para a fusão de base de dados heterogêneos. Esse processo de integração das bases consiste em criar modelos de esquema de base de dados, gerar os modelos de esquema e finalmente gerar o *scrip* SQL.

O esquema de base de dados é composta por entidades, atributos e relacionamento que fazem parte do banco de dados a ser integrado. Através das definições de transformações escritas em ATL o *integrated database model* se transformasse em *databse model merging* o que é exatamente a estrutura do esquema de base de dados, esse modelo irá ser transformado através das definições de transformação em um PSM, no caso um *script* SQL.

3.2 Abordagens para o desenvolvimento de aplicações sensíveis ao contexto

Nesta seção, alguns trabalhos que abordam o desenvolvimento de aplicações sensíveis ao contexto são analisadas.

3.2.1 A Framework for Developing Mobile, Context-aware Applications [47]

O trabalho de BIEGEL [47], trata-se de um *framework* que facilita o desenvolvimento de aplicações sensíveis ao contexto. Eles desenvolveram um modelo de objeto para desenvolver aplicações sensíveis ao contexto em ambiente móvel ad-hoc , que define abstrações de *software* para sensores e atuadores. O trabalho fornece um *framework* para especificação da regra de produção de comportamento conduzindo objetos sencientes que possuem uma série de características que são importantes em ambiente de computação ubíqua [47].

Segundo BIEGEL [47] o modelo de objeto sensível fornece uma abordagem sistemática para o desenvolvimento de aplicações sensíveis ao contexto em ambientes adhoc móveis, suportando os aspectos importantes de fusão de sensores, a extração de contexto e raciocínio. O *framework* que suporta o desenvolvimento de aplicações da seguinte forma:

- fornece abstrações para sensores e atuadores, aliviando assim o desenvolvedor do ônus do baixa nível de interação com vários dispositivos de hardware;
- fornece um mecanismo probabilístico para fundir fragmentos multimodais de dados do sensor em conjunto a fim de obter informações de contexto de nível superior;
- fornece uma abordagem eficiente para o raciocínio inteligente baseado em uma hierarquia de contextos;
- fornece um mecanismo de comunicação baseada em evento para interação entre os sensores, objetos e atuadores;

- fornece uma ferramenta de programação visual de fácil acesso para aplicações em desenvolvimento, reduzindo a necessidade de escrever código.

A Figura 12 mostra essencialmente que , este objeto é uma entidade encapsulada com suas respectivas interfaces, sensores e atuadores [47]. As ações são controladas com base na entrada da informação obtida pelo sensor de acordo com a lógico de controle interno. Esta lógica consiste em filtragens de eventos, fusão de sensores e inferência inteligente. Segundo o autor o sensor de objeto sensíveis é definido como uma entidade que produz eventos de *software* em reação ao mundo real. Os atuadores são os elementos que consomem os eventos de *software* e reagem tentando mudar o estado do mundo real.

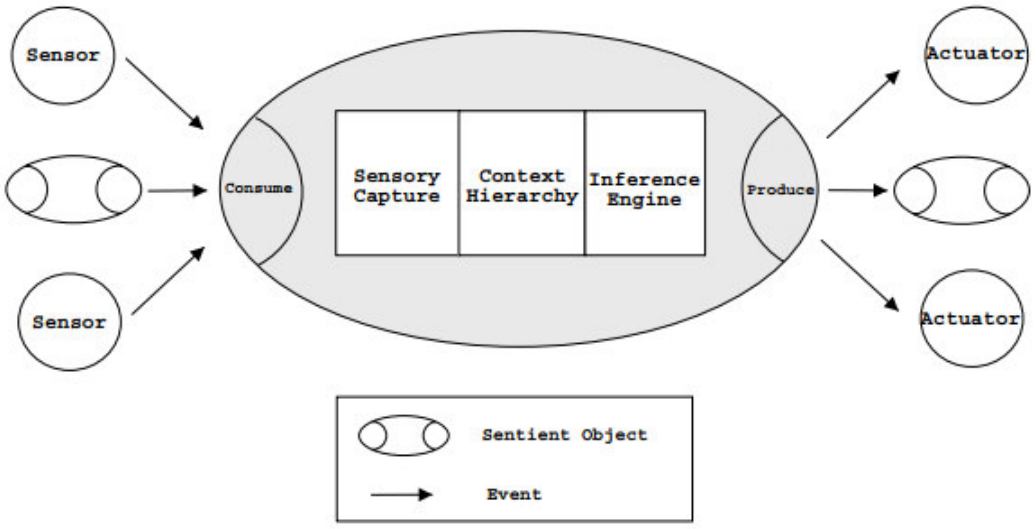

Figura 12: Modelo de Objeto Sensível [47]

O modelo de objetos sensíveis são compostos por 3 principais componentes internos tais como :

- a captura sensorial e o âmbito de fusão que captura os objetos e realiza a fusão e gerenciamento das incertezas dos dados utilizando de técnicas Baysesianas;
- hierarquia de contexto que encapsulam conhecimento sobre ações a serem tomadas e possíveis situações futuras;

- fusão de sensores e a hierarquia contexto, é realizado uma fusão de sensores ao nível de um contexto dentro da hierarquia de contexto.

o trabalho de BIEGEL [47] fornece uma abordagem sistemática para o desenvolvimento de aplicações sensíveis ao contexto incluindo a capacidade de fundir fragmentos de contexto e lidar com incertezas de forma probabilística [47].

3.2.2 A Middleware for Building Context-Aware Mobile Services [76]

O presente artigo escrito por GU et al[76], descreve um *middleware* orientada a Serviços Sensíveis ao contexto (SOCAM), para a prototipagem rápida de serviços móveis. A proposta do trabalho adota " uma abordagem orientada a serviços para construir o *middleware* que suporte tarefas, incluindo a aquisição de descobertas, interpretação, acesso a vários contextos e interoperabilidade entre diferentes sistemas sensíveis ao contexto " [76].

O *middleware* proposto no trabalho de [76] faz a conversão dos espaços físicos onde os contextos são adquiridos a partir de um espaço semântico. Estes contextos podem ser compartilhados e acessados por serviços sensíveis ao contexto. A Figura 13 mostra que ele consiste de provedores de contexto, interpretador de contexto, contexto de banco de dados, serviço de localização do serviço sensível ao contexto em serviços móveis.

Os provedores de contexto fornecem um contexto abstrato que separa o contexto de baixo nível de detecção para o de manipulação de alto nível. O interpretador de contexto tem a atuação parecida com o provedor de contexto onde proporciona contextos de alto nível ao interpretador contextos de baixo níve [76].

O serviço de localização de serviço permite localizar os diferentes provedores de contexto, este pode carregar as ontologias de contexto que estão armazenadas nas instâncias do banco de dados e o próprio contexto que está inserido nos provedores de contexto. Os serviços móveis sensíveis ao contexto são aplicações e serviço que fazem uso de diferentes níveis de contexto e aplicam a correspondência semântica para descobrir qual o provedor de contexto será necessário para um dado contexto [76].

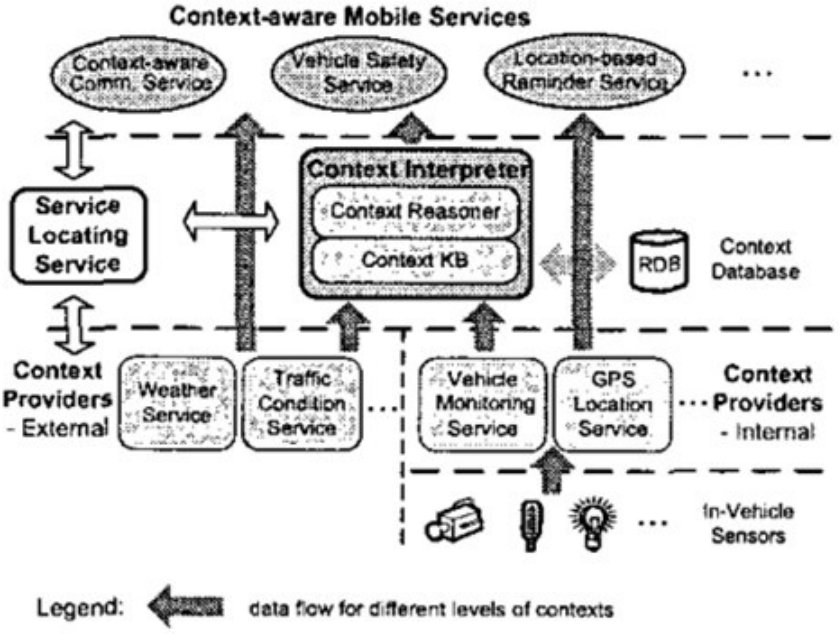

Figura 13: Arquitetura SOCAM [76]

O interpretador de componentes também pode registrar-se para o Serviço de Localização de serviço ou de outros mecanismos de descoberta de serviço, para que possam ser descobertos e acessados por outros sistemas sensíveis ao contexto [76].

Este trabalho mostra uma *middleware* baseada em aquisição de contexto e descoberta de contexto assim como a sua divulgação, um *middleware* orientado a serviço que permite a prototipagem de serviços móveis sensíveis ao contexto.

3.2.3 Mobile Gaia: A Middleware for Ad-hoc Pervasive Computing [77]

O trabalho de SHIVA el al [77] mostra um middleware baseado em serviços que integra recursos de vários dispositivos. Este *midleware* gerencia diversas funções, tais como a formação e manutenção das coleções de dispositivos, a partilha de recursos entre os dispositivos e permite interações de serviço sem costura. Ele também fornece uma estrutura de aplicativo para desenvolver aplicativos para a coleção dispositiva. A estrutura do aplicativo decompõe a aplicação em componentes menores que podem ser executados em dispositivos diferentes nesta coleção, sua estrutura pode ser vista na Figura 14 [77].

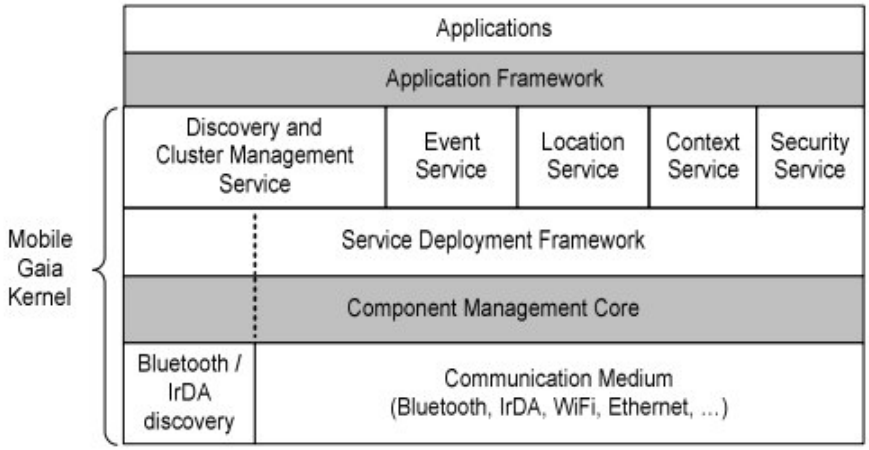

Figura 14: Arquitetura Gaia [76]

O projeto de [77] foi desenvolvido para redes ad-hoc e computação pervasiva, consiste em um conjunto de serviços básicos que administram um *cluster*. O serviços que gerenciam o *cluster* descobre os diversos dispositivos em sua vizinhança, negocia com os dispositivos para ingressar no *cluster* e gerencia a composição do mesmo. O serviço de segurança fornece a autenticação e controle de acesso ao *cluster*, e o *framework* de implementação do serviço atua como um recipiente para esses serviços [77].

Os serviços do Mobile Gaia são implementados pelo *framework* que gerencia a instalação de novos componentes de serviço, carga e descarga e a remoção dos componentes quando não são mais necessários. Em geral este projeto facilita a integração de recursos de um conjunto de dispositivos pessoais e fornece uma interface de programação comum para programar *clusters*. O *middleware* também permite suportar um *framework* de aplicativo que permite que o aplicativo seja distribuído em vários outros dispositivos [77].

3.3 MDE e aplicações sensíveis ao contexto

Nesta seção, alguns trabalhos que abordam o desenvolvimento de aplicações sensíveis ao contexto baseados em MDE serão apresentados.

3.3.1 A Model-Driven Approach to Generate Context Aware Applications [78]

Segundo DUARTE et al [78] propõem uma Linguagem Específica de Domínio Visual para modelagem de informações contextuais. Segundo DUARTE et al [78], o LOCAM pode ser definida da seguinte forma:

> "*é uma infraestrutura de gerenciamento de contexto voltada para dar suporte a aplicações sensíveis ao contexto em dispositivos móveis, por exemplo: aplicações de anotações contextuais, fotos etc*"DUARTE [78].

A Figura 15 mostra como deverá ocorrer o processo de modelagem e geração de código seguindo a proposta do artigo, ou seja utilizando o DLS LoCCAM. O metamodelo DSL-LoCCAM é baseado em Ecore e criado a partir do EMF. Segundo o autor a ferramenta visual foi criada com base no *framework* GMF *(Graphical Modeling Framework)*, junto a ferramenta Eugenia. As regras de transformações foram realizadas através do xPand, Linguagem de transformação que é baseada em EMF [78].

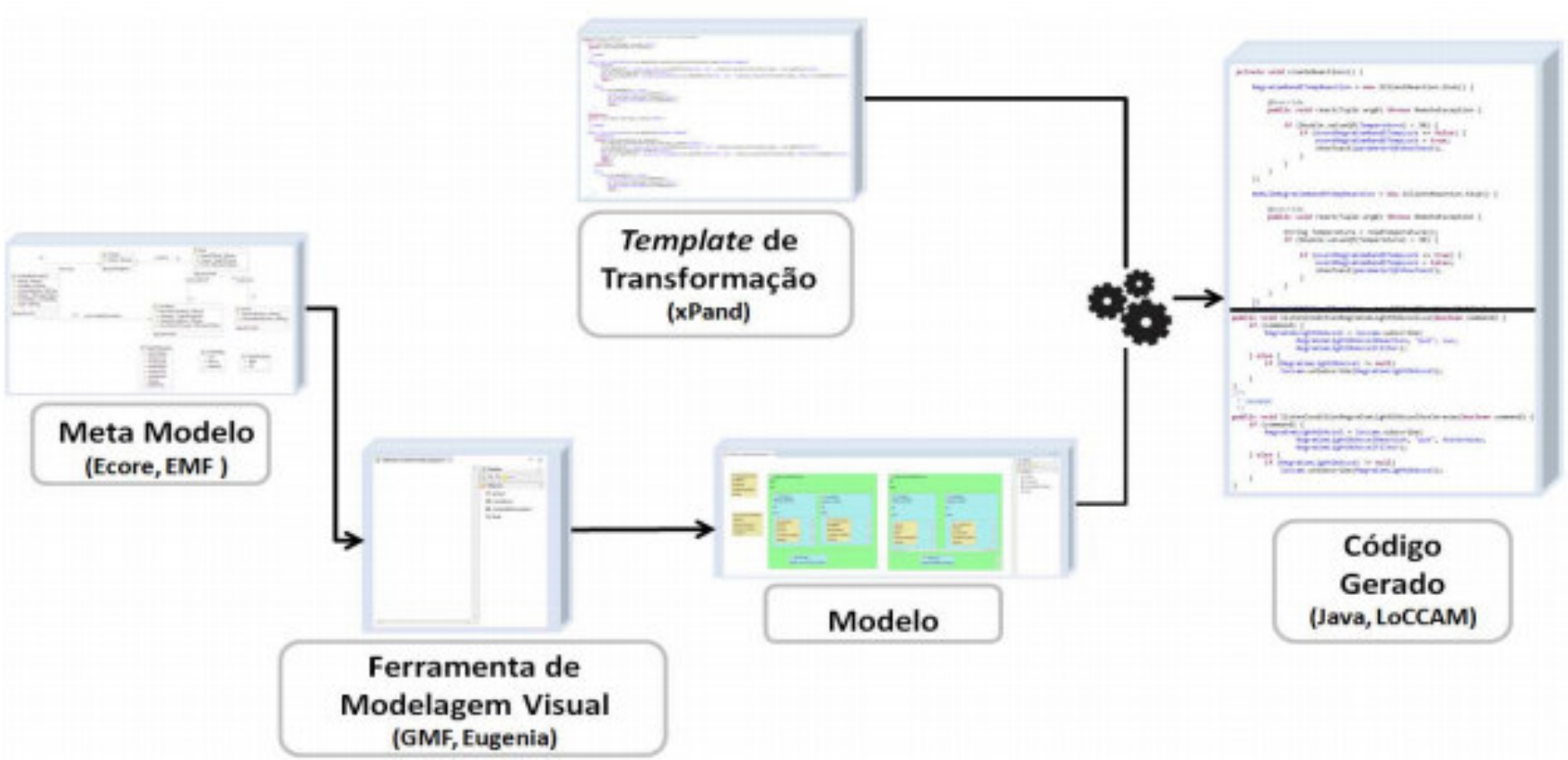

Figura 15: Processo LoCCAM [76]

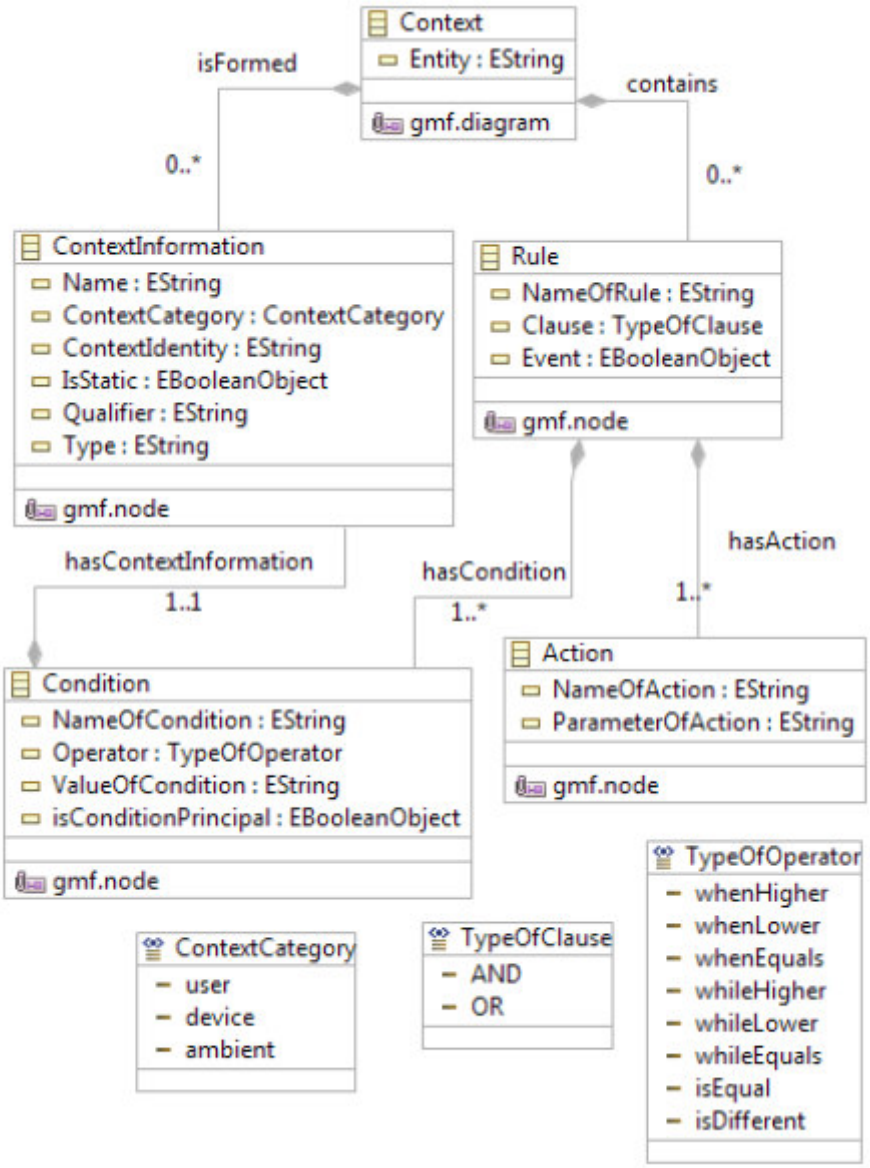

Figura 16: Metamodelo do DSL-LoCCAM [78]

O artigo apresenta um metamodelo (Figura 16) proposto para a DSL-LoCCAM que engloba os contexto existentes em uma aplicação, composto com regras, informações contextuais, agregação de condições e ações. A sintaxe é simples, dentro do *ContextInformation* são especificadas as informações contextuais. O *ContextIdentity* explora as informações contextuais para que o *middleware* possa interpretar o que se deseja fazer com a informação. Em *Rule* indica as regras contextuais que serão criadas no modelos. Em *TypeOfOperator* oferece uma variedade de informações que deverão ser comparadas e em *Action* especifica o que deverá ser realizado pelo evento ou estado quando a condição for satisfatória [78].

Segundo DUARTE et al [78] o trabalho tem como principal objetivo otimizar o uso do LoCCAM pelo desenvolvedor,abstraindo a necessidade de adquirir conhecimentos específicos sobre a sintaxe do *midleware*. Isto automatizaria a escolha de possíveis componentes que correspondem as informações contextuais [77].

3.3.2 Model Driven Development of Context Aware Software Systems [75]

No trabalho de SINDICO [75] , apresenta-se uma linguagem específica de domínio para modelagem que permite aos engenheiros de *software* lidar com *context-aware* na fase de um projeto de sistema, para isto foi criado o CAMEL um metamodelo capaz de instanciar o modelo conceitual para uma percepção de contexto.

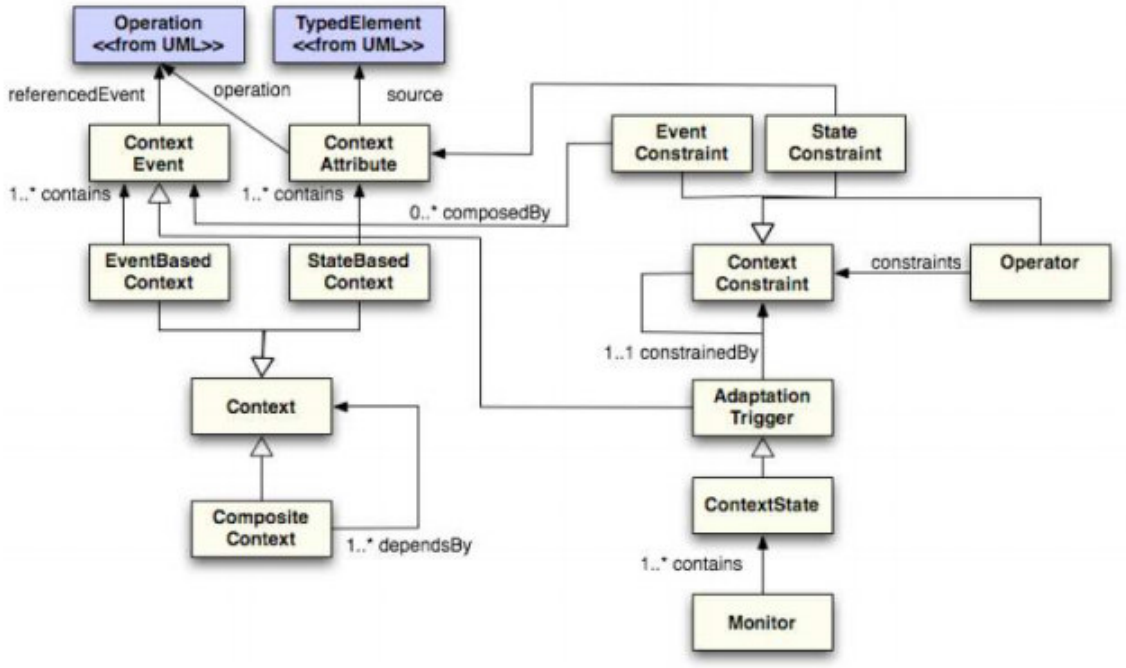

Figura 17: Metamodelo CAMEL - context sensing [75]

A Figura 17 e a Figura 18 ilustram o CAMEL [75],na qual a preocupação de consciência de contexto é tratada por meio de três partes separadas: senso de contexto, desencadeamento de adaptação de contexto e adaptação de contexto. Senso de contexto ou *context sinsing* engloba o conjunto de atividades destinadas a recuperar informações contextuais de sensores físicos ou lógicos. O desencadeamento da adaptação de contexto é definido pelo conjunto de atividades que avaliam continuamente informações contextuais detectadas e, dependendo de certas condições, desencadeiam a ativação de mecanismos de adaptação. A adaptação de contexto é finalmente definida pelo conjunto de mecanismos de adaptação que podem ser acionados e então ativados em resposta aos gatilhos de adaptação ao contexto.

CAMEL oferece duas construções de modelagem de informações contextuais, o *StateBasedContext* e *EventBasedContext*, o primeiro é o contêiner para informações contextuais estáticas que consiste em um conjunto de atributos, que são representadas por *ContextAttribute*, relevantes para uma determinada definição de contexto.

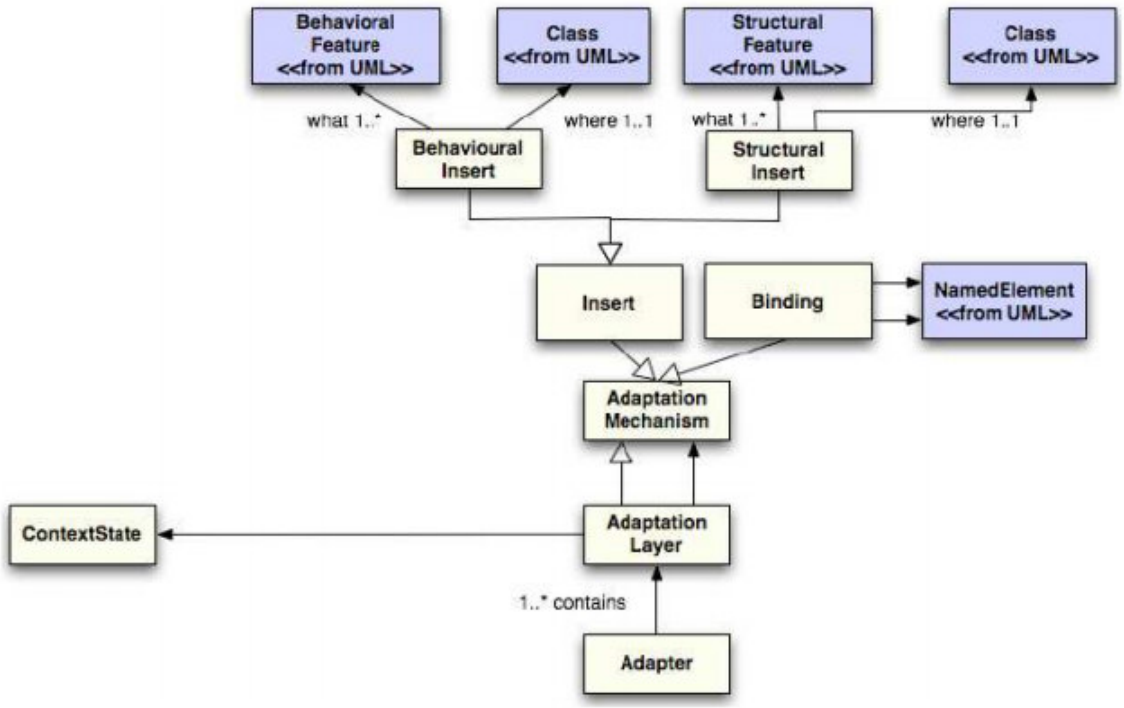

Figura 18: Metamodelo CAMEL - adaptation triggering [75]

ContextAttribute é caracterizado por um nome e uma relação de origem com a UML: *TypeElement*, representa o recurso estrutural do sistema de destino , tal como: um atributo, uma associação, parâmetro etc. o *ContextEvent* é um conjunto de eventos que devem ser relevantes para uma dada definição de contexto. O *Monitor* representa o contêiner para gatilhos de adaptação logicamente relacionados. Um gatilho pode ser considerado com uma condição de estado pela informação contextual que é recuperada baseada em estados e eventos que são monitorados pelo *Monitor* [75].

A Figura 18 descreve melhor a adaptação de contexto. Segundo [75] esta adaptação pode ser definida como um conjunto de mecanismos de adaptação que podem ser acionados e então ativados durante atividades de monitoramento de contexto ara reagir adequadamente a mudanças de contexto. Assim, [75] propõe que a linguagem CAMEL possa realizar através de dois mecanismos *:context aware bindings* e *context aware inserts* a adaptação contextual.

Segundo SINDICO [75], a camada de adaptação é definida como uma construção que atua como um recipiente para mecanismos que estão interligados logicamente. Assim,

quando esta camada é ativada, os mecanismos de adaptação são ativos e a adaptação que é desejada é introduzida.

Na linguagem CAMEL, os adaptadores são aquelas entidades que atuam como contêiner para camadas de adaptação logicamente relacionadas. Os adaptadores recebem sinais pelos monitores e ativam/desativam suas camadas de adaptação dependendo dos estados de contexto que estão atualmente ativos [75].

O presente trabalho aborda um *framework* utilizando MDD para *context-aware* e uma linguagem adaptada para modelagem de comportamentos cientes de contexto *context-aware).*

3.4 Síntese

Esta Capítulo apresentou o Estado da Arte refente as abordagens que dão suporte ao desenvolvimento de sistemas sensíveis ao contexto.

Pode ser verificado nos trabalhos [42, 41, 76, 77, 75, 47, 78], apresentados neste Capítulo, métodos e abordagens que visam facilitar o desenvolvimento de sistemas de *software*. Sete trabalhos foram citados, foram descritas suas carateristas e seus objetivos principais, além de abordar as técnicas utilizadas pelos respectivos trabalhos em seus *frameworks* ou *midlewares* citados.

4 FRAMEWORK BASEADO EM MDE E WEAVING PARA SUPORTAR O DESENVOLVIMENTO DE SISTEMAS DE SOFTWARE SENSÍVEIS AO CONTEXTO

Neste capítulo, um *framework* para suportar o desenvolvimento de sistemas sensíveis ao contexto será apresentado. Este *framework* é baseado em MDE e *Weaving*, permitindo a criação de modelos e o entrelaçamento dos mesmos para a geração de modelos mais específicos. Uma metodologia de aplicação do *framework* proposto é apresentada, e os metamodelos são fornecidos ou adaptados para suportar a criação de modelos.

BÉZIVIN et al [37], propõem um processo de desenvolvimento em Y, contendo PIM, PDM, Weaving e PSM para suportar o desenvolvimento de sistemas de *software*. Nosso trabalho de pesquisa, propõe um *framework* de desenvolvimento baseado em Y, porém, utilizando o entrelaçamento entre PIM e PDMs de forma paralela, i.e. um único modelo de *Weaving* que relaciona um PIM e vários PDMs. O entrelaçamento de modelos de forma paralela agiliza o desenvolvimento de sistemas de *softwares*, uma vez que, ao realizar o entrelaçamento tradicional (um entrelaçamento por vez) se faz necessário utilizar vários modelos de *Weaving* contendo cada um os entrelaçamentos entre um único PIM e um único PDM.

Para exemplificar a proposta, mostramos a Figura 19 que retrata um modelo de como funciona o entrelaçamento de forma serial e a Figura 20 mostra como é o funcionamento de forma paralela.

Na forma serial (Figura 19) o PIM é entrelaçado com cada PDM, ou seja, um por vez. Cada *Model Weaving* é gerado a partir do entrelaçamento dos elementos do PIM e do respectivo PDM. Após esse entrelaçamento o PSM do modelo de *Weaving* é criado. O primeiro PSM então foi criado, desta forma volta-se a um novo entrelaçamento entre o PIM e o segundo PDM que deverá gerar um novo *Model Weaving* e então a partir deste criar um novo PSM.

Na forma paralela (Figura 20), forma utilizada neste trabalho, o entrelaçamento entre o PIM e PDMs é realizado em uma única etapa, gerando o *Model Weaving* com os elementos dos PDMs e PIM e posteriormente gerando apenas um único PSM.

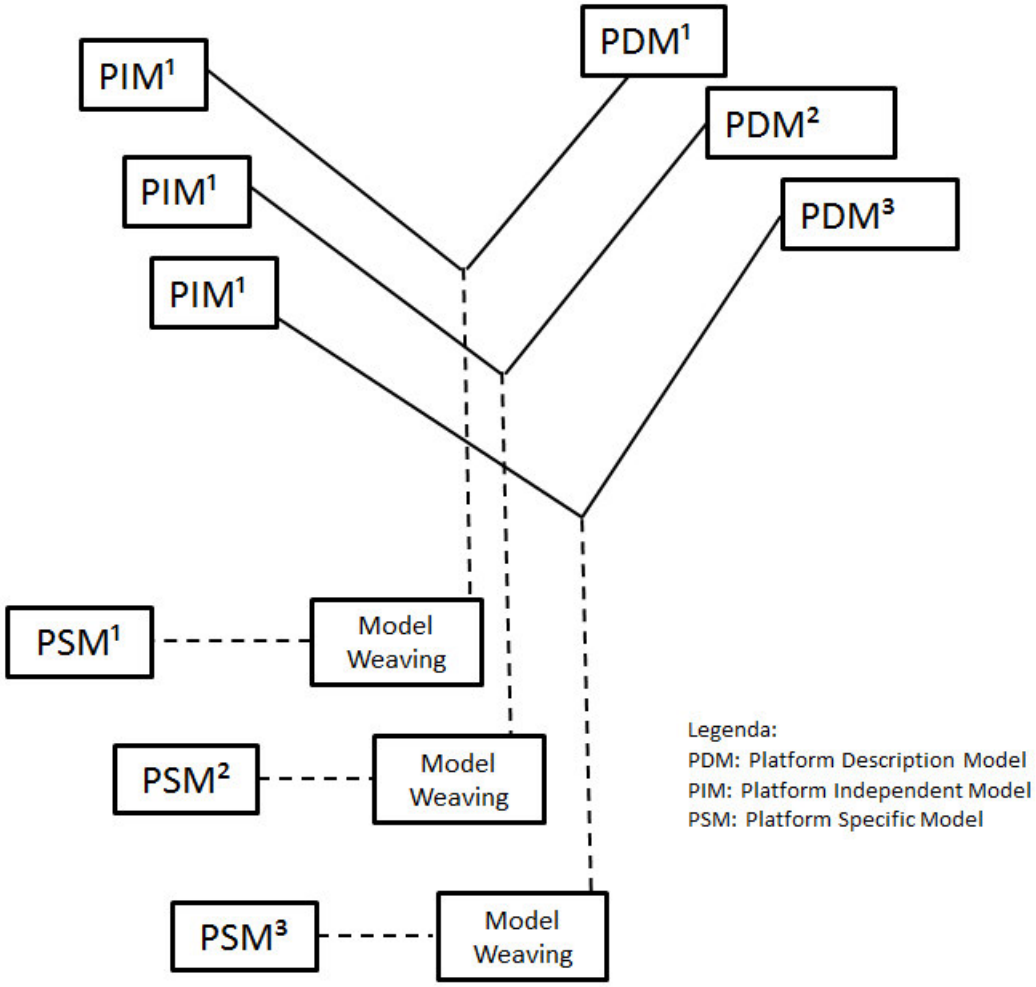

Figura 19: Técnica de *Weaving* - Forma Serial

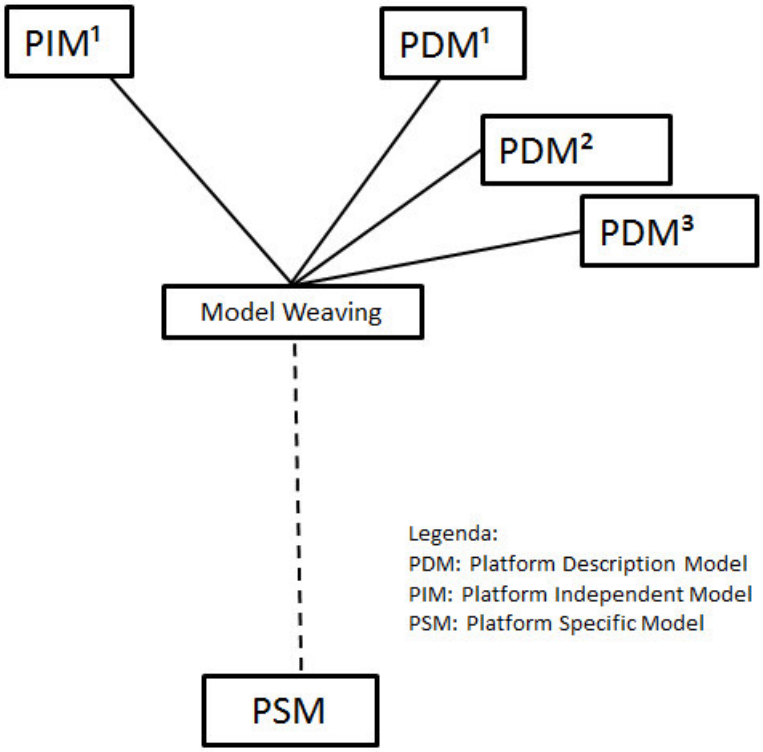

Figura 20: Técnica de *Weaving* - Forma Paralela

4.1 Arquitetura do framework proposto

A Figura 21 apresenta a arquitetura do *framework* proposto que é formado pelo: é formado por modelos: o PIM conforme o metamodelo UML, o PDM1 conforme o metamodelo de Contexto, o PDM2 conforme o metamodelo de Segurança e o PDM3 conforme o metamodelo de Sistemas Distribuídos, também em sua arquitetura contamos com o Modelo *Weaving* que está conforme ao metamodelo de *Weaving*. É importante salientar que este *framework* é passível a extensão, ou seja, não se limita unicamente em três PDMs. Assim, caso o desenvolvedor necessite de demais plataformas o mesmo pode expandir o *framework*, inserindo estas plataformas na estrutura.

O *framework* apresentado na Figura 21 possui os seguintes elementos:

- na primeira etapa, pode-se constatar que possuímos todas as criações dos modelos necessários, tais como PIM e PDMs. Nesta etapa também está incluso a ferramenta criada e aperfeiçoada para essa pesquisa. O processo que ocorre nesta etapa é proveniente do entrelaçamento realizado entre os PIM e PDMs por meio

da *Weaving Tool* que por sua vez gera o modelo de *weaving*;

- na segunda etapa, por meio do modelo de *weaving* e após o entrelaçamento entre os nós deste modelo, o Intermediate Model é criado. *Intermediate Model* será o modelo de entrada para a realização das transformações a fim de gerar o PSM abstrato que está em conformidade com o *Web Service Metamodel*;

- Em seguida, na terceira etapa, o motor de transformação tem como entrada o PSM abstrato onde ocorre a transformação de modelo a modelo gerando assim o PSM Concreto que é conforme o *Java Metamodel*;

- Por fim, e última etapa, outro mecanismo de transformação tem como entrada o PSM Concreto gerando como saída o Source Code.

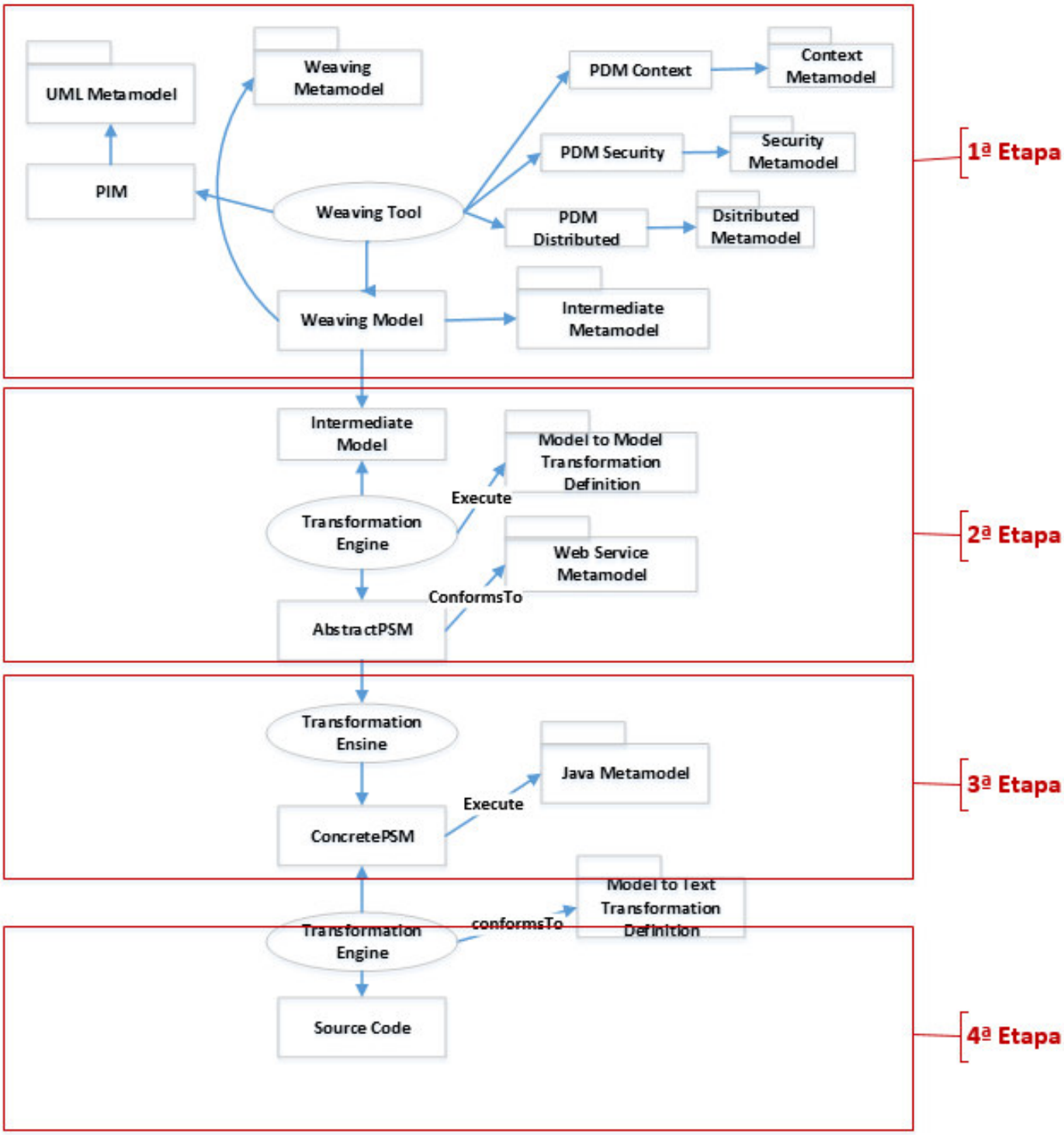

Figura 21: Um *framework* baseado em MDE e *Weaving* para suporte ao desenvolvimento de sistemas sensíveis ao contexto.

4.2 Metodologia para aplicação do framework proposto

A metodologia para aplicação do *framework* proposto é apresentada na Figura 22. Esta metodologia é composta pelas seguintes etapas:

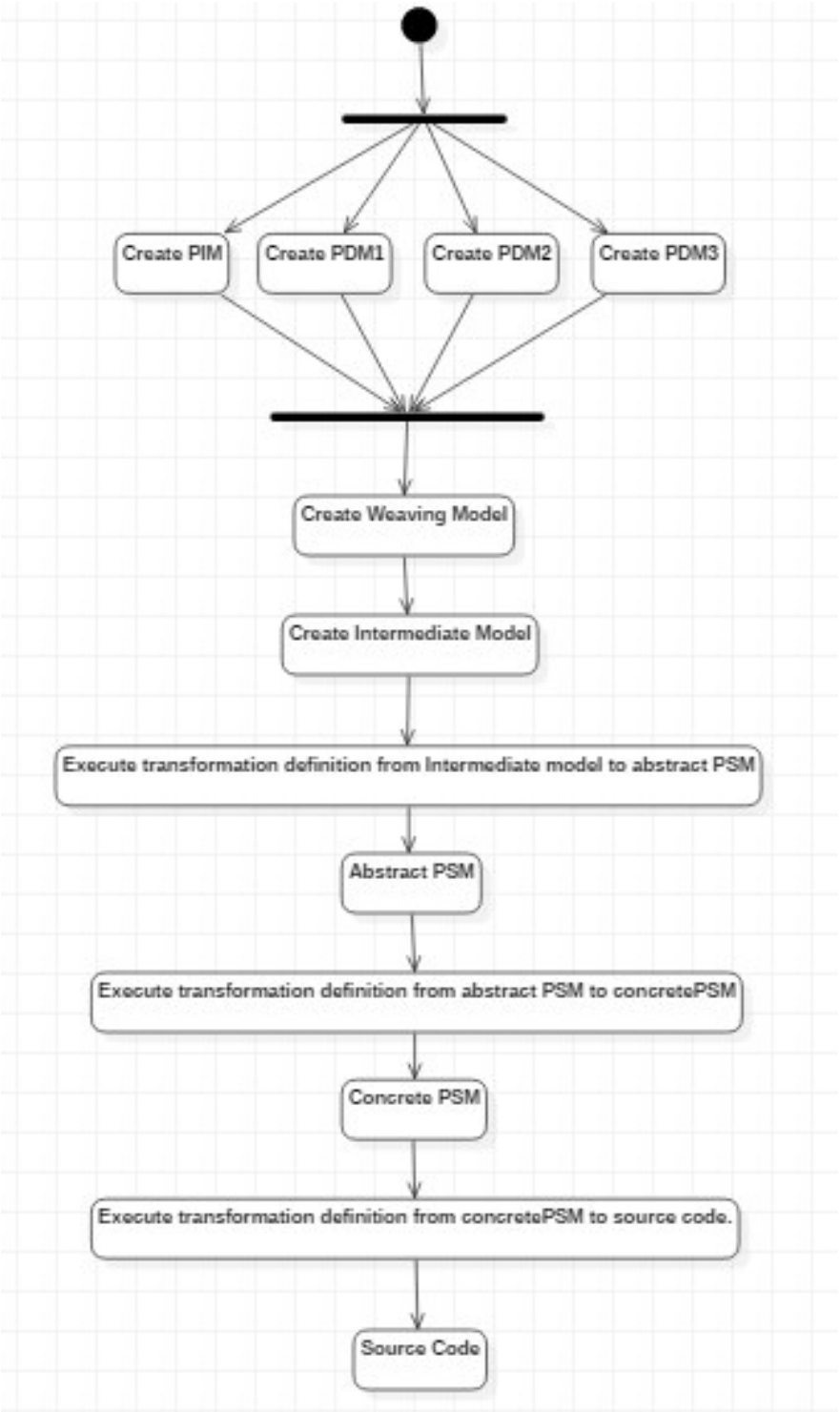

Figura 22: Uma metodologia para para aplicação no *framework* proposto

- **Create PIM:** Inicialmente, criamos o PIM, conhecido também como modelo de negócios;
- **Create PDM:** cria-se o PDM, que deverá ser conforme o seu respectivo metamodelo;
- **Create Weaving Model:** nesta fase, o modelo *Weaving* é criado e tem como propósito te relacionar os modelos PIM e PDMs;
- **Create Intermediate Model:** modelo que possui a fusão dos dos modelos PIM, PDMs e Modelo *Weaving*;
- **Execute transformation definition from Intermediate Model to Abstract Model:** mecanismo de transformação, onde ocorre a transformação de modelo para modelo;
- **Abstract PSM:** fase que tem o intuito de incluir informações específicas que não estejam presentes no modelo intermediário, sejam informações do PIM ou do PDM;
- **Execute transformation definition from abstract PSM to concrete PSM:** esta fase tem o intuito de gerar o Concrete PSM que será o PSM utilizado na transformação para o Código fonte;
- **Concrete PSM:** PSM gerado que será o modelo de entrada para a transformação de Modelo e Texto;
- **Execute transformation definition from concrete PSM to souce code:** mecanismo de transformação, onde ocorre a transformação de modelo para texto;
- **Source Code:** esboço do programa em código fonte criado.

4.3 Metamodelos propostos

Nesta seção, os metamodelos propostos que compõem o *framework* serão apresentados.

4.3.1 Metamodelo de Weaving

Para se criar o modelo *Weaving* foi necessário usar o metamodelo de *Weaving* visto na Figura 24 e Figura 23, que possui os elementos necessários para fazer o entrelaçamento entre PIM e PDMs. O metamodelo da Figura 23 é baseado no trabalho de LOPES, HAMMOUDI et al. [80]. Fizemos uma evolução deste metamodelo, inserindo mas duas classes que representam as plataformas que serão trabalhadas. Este metamodelo é descrito como a seguir:

- **Historic:** elemento que armazena atributos: update, node, servion e timestamp e todos os históricos contendo configurações de tecelagem (WeavingDefinition);
- **ModelHandlers:** funciona como um ponteiro para relacionar modelos;
- **WeavingNode:** responsável por relacionar ou mais elementos Indicados por *LinkLeft, LinkRight, LinkRightContext, LinkRightDisitrbuted e ElementModel*;
- **Handlers:** elemetntos de um metamodelo que poderá se relacionar com elementos de outros determinado metamodelo [69];
- **LinkLeft:** modelo do lado esquerdo, este será o modelo PIM que fará parte do processo de criação do modelo de *weaving*;
- **LinktRight:** primeiro modelo do lado direito, responsável por mapear os aspectos do metamodelo de segurança, este corresponderá aos aspectos do PDM;
- **LinkRightContext:**segundo modelo à direita, Responsável por mapear aspectos do metamodelo *contextaware*;
- **LinkRightDistributed:**terceiro modelo no lado direito, responsável por mapear aspectos do metamodelo de sistemas distribuídos.

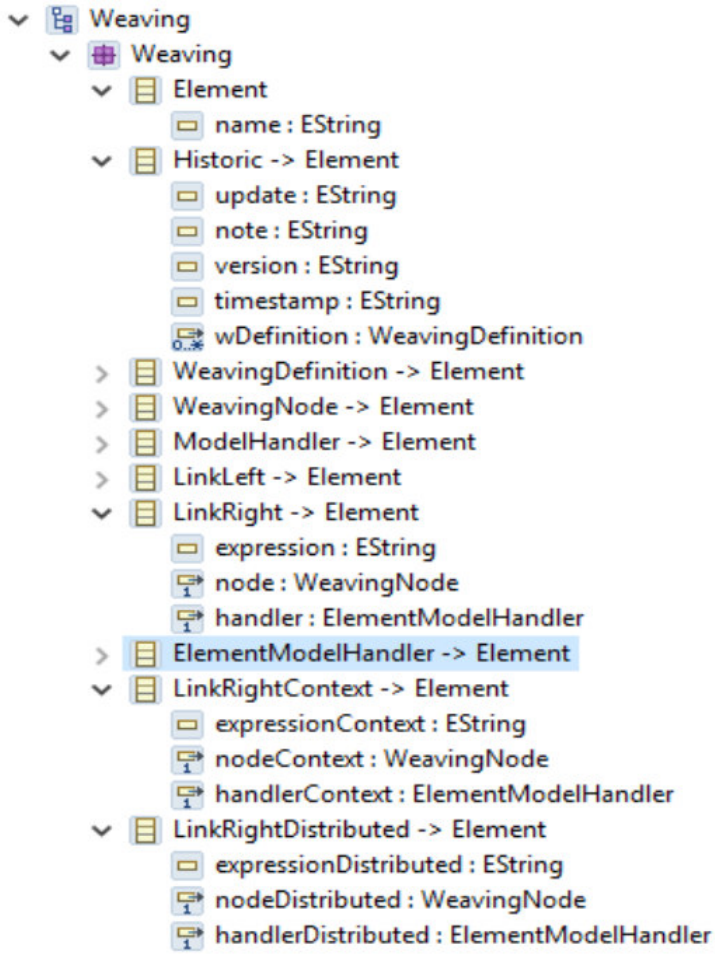

Figura 23: Metamodelo de Weaving baseado em [80] (forma de árvore).

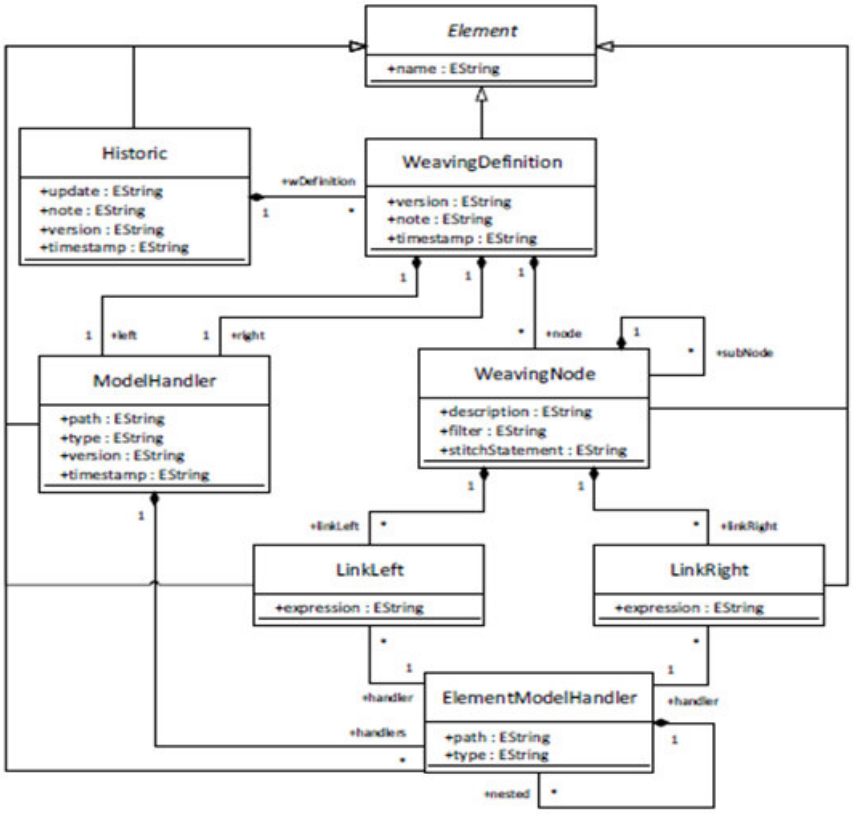

Figura 24: Metamodelo de Weaving baseado [80] (conforme a notação de UML).

4.3.2 Metamodelo para PDM com aspectos de segurança

O metamodelo escolhido como nosso PDM de segurança foi baseado em [69]. Este foi escolhido por possuir aspectos de segurança com o foco central em autorização e autenticação, o que por sua vez, corrobora com elementos de mecanismos de controle de acesso baseado em funções [69]. Esses mecanismos são importantes para que o sistemas faça autenticação dos dados do usuário, o que consequentemente poderá auxiliar o sistema a saber quem está o usando.

É importante salientar que este metamodelo também possui elementos baseados em [79] e [83]. Na Figura 25 e Figura 26 podemos detalhar os elementos primordiais para a ação do *framework* aqui proposto. Os elementos do metamodelo do PDM de segurança são descritos conforme a seguir:

- **RoleSpace:** Container para funções (*Rules*) que determinam as ações realizadas pelo usuário [69];

- **User:** elemento responsável por conter propriedade e operação;
- **Permission:** elementos que têm restrições das ações realizadas pelos recursos aos quais o usuário tem acesso.

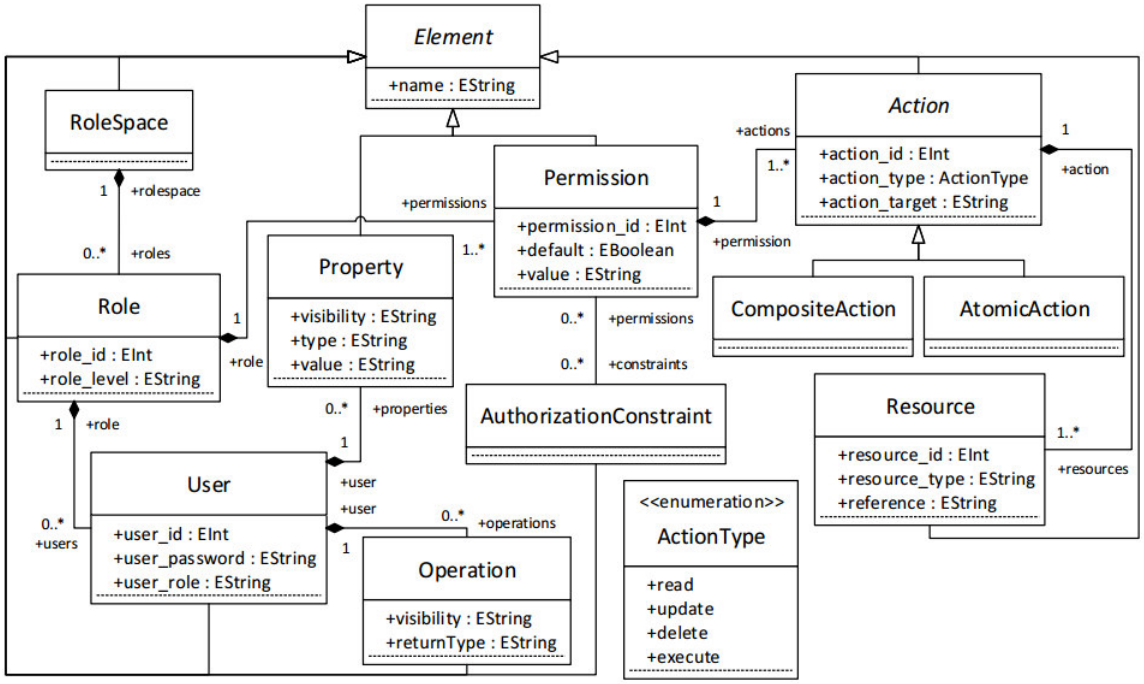

Figura 25: Metamodelo com aspectos de segurança baseado em [69], [79], [83]

Figura 26: Metamodelo com aspectos de segurança baseado em [69], [79] e [83]

4.3.3 Metamodelo para PDM com aspectos de Contexto

O metamodelo para descrever sistemas sensíveis ao contexto é apresentado na Figura 27 que é baseado no trabalho de Taconet, Zazia et al [82]. A Figura 27 apresenta o metamodelo de contexto, onde Segundo Taconet, Zazia et al [82] a meta-classe *Context Aware System* é o ponto de partida deste metamodelo, classe que deve gerenciar todo o ocorrido dentro do metamodelo. A parte esquerda do meta-modelo define as entidades e como são relacionadas, assim como, seus *observable*. Os *Interpreted observable* e os *AdaptationSituations* são os elementos responsáveis pelas mudanças ocorridas no sistema. E na ultima parte ao lado direito do meta-modelo podem ser definidos os contratos de *context aware system*.

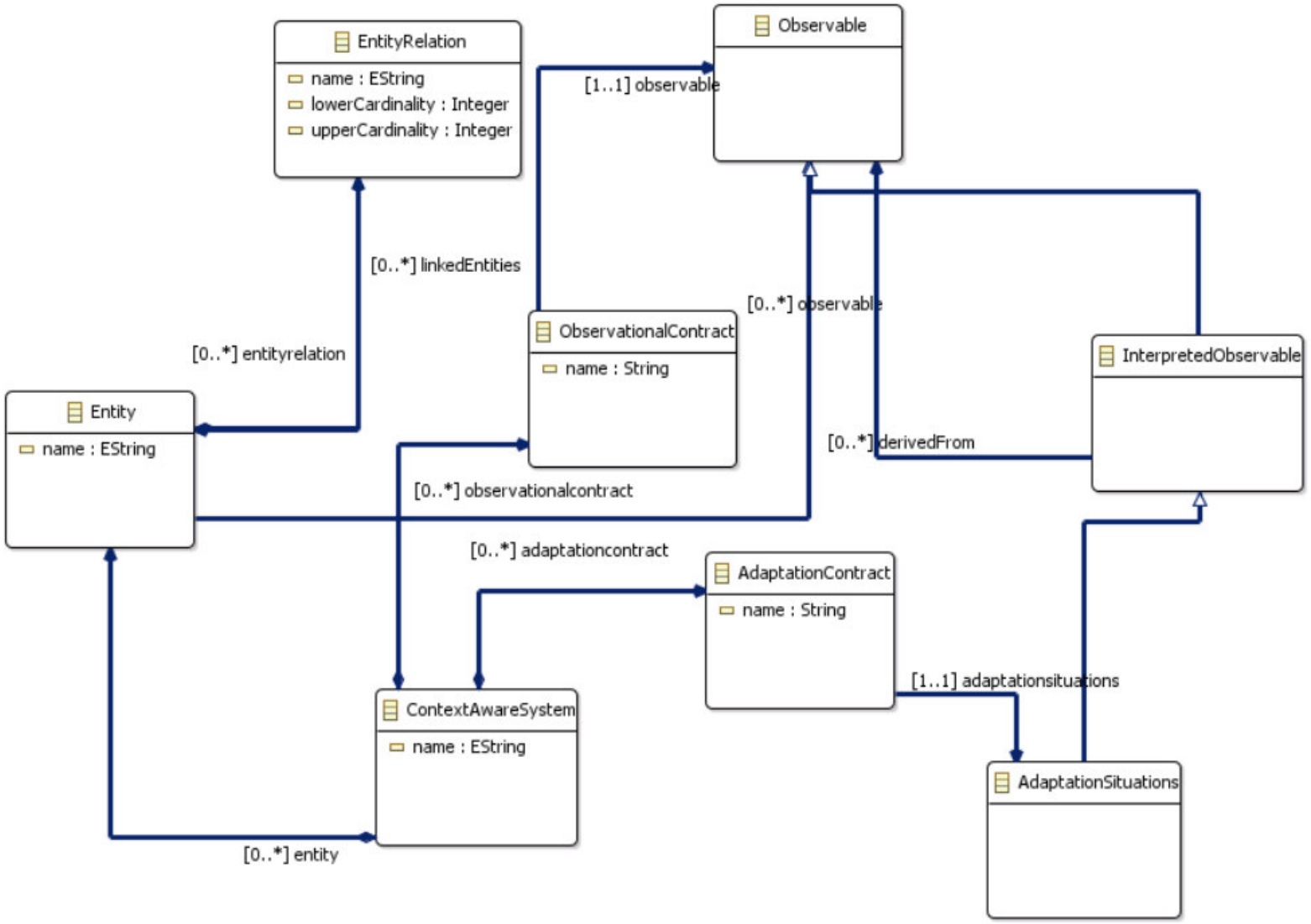

Figura 27: Metamodelo com aspectos de *Context Aware Systems* baseado em [82]

4.3.4 Metamodelo para PDM com aspectos de Sistemas Distribuídos

O metamodelo de Sistemas Distribuídos foi baseado no EDOC (*Enterprise Distributed Object Computing*), o metamodelo pode ser visto na Figura 28. O metamodelo para sistemas distribuídos contém os seguintes elementos:

- **ProcessComponent:** representa o contrato para um componente que executa ações - "faz algo". Um ProcessComponent pode realizar um conjunto de portas para interação com outros *ProcessComponents* e pode ser configurado com propriedades"[71];

- **Ports:** é o conjunto de portas no *ProcessComponent*. Cada porta fornece um ponto de conexão para interação com outros componentes ou serviços e realiza um protocolo específico. O protocolo pode ser simples e usar um "*FlowPort*"ou o protocolo pode ser complexo e usar um "*ProtocolPort*"ou um "*OperationPort*". Se permitido por seu protocolo, uma porta pode enviar e receber informações [71];

- **Node:** o nó é um elemento abstrato que especifica algo que pode ser a fonte e / ou Alvo de uma conexão ou transição e assim ordenados dentro do processo coreografado;

- **Choreography para port activites**: envio ou recebimento de mensagens ou iniciação de subprotocolos;

- **Entity Component:**autorização de Campos de interfaces que necessitam de retorno;

- **OperationPort:**mensagens para os seus próprios processos componente;

- **Property Definitions:** define um parâmetro de configuração do componente, com pode ser definido quando o componente é usado [71];

- **MultiPort:**é um agrupamento de portos cujas ações estão ligadas. Informações devem estar disponíveis em todos os subports do MultiPort para qualquer ação ocorrer em um componente anexado "[71].

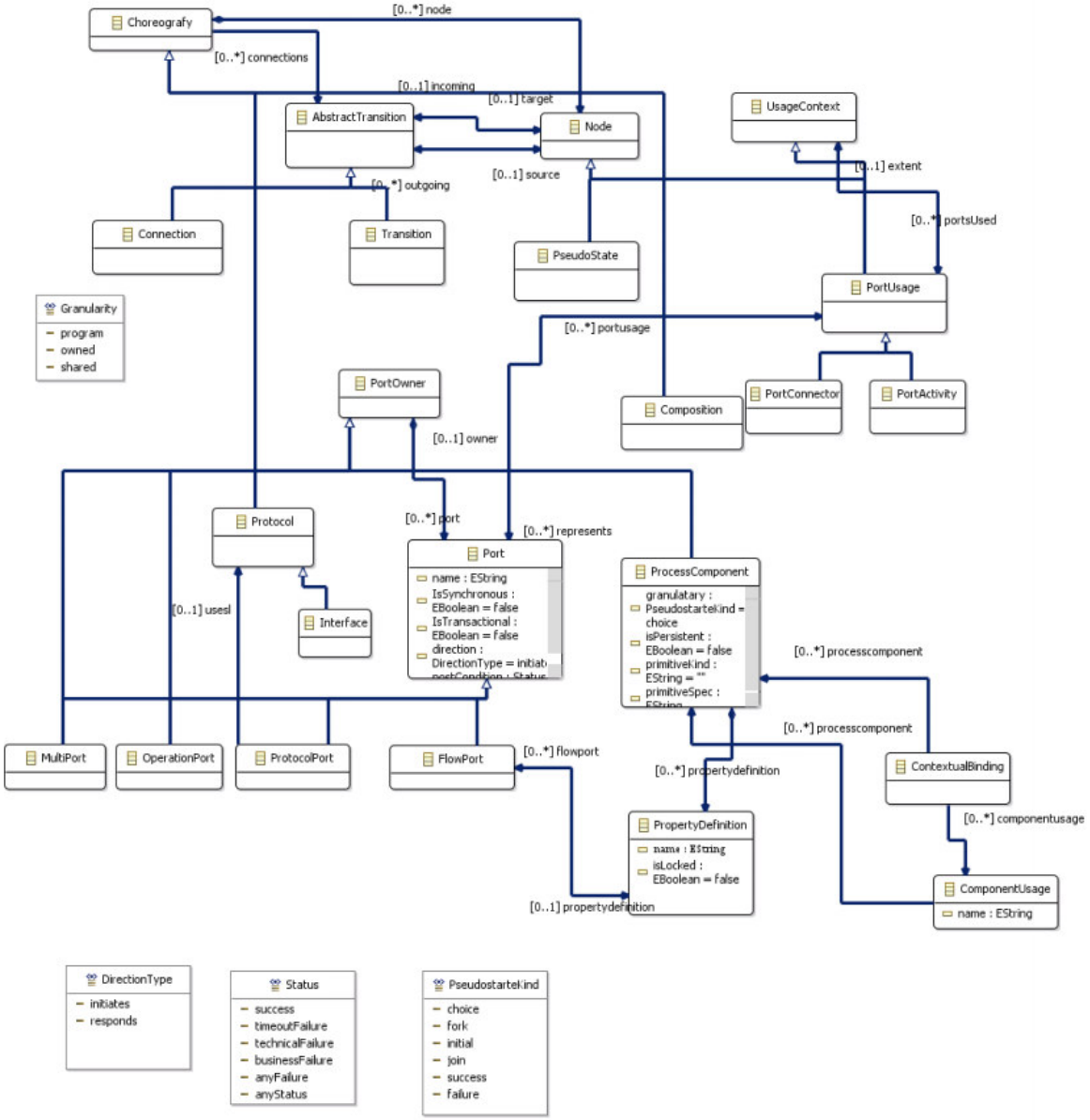

Figura 28: Metamodelo de Sistemas Distribuídos [71]

4.3.5 Metamodelo Intermediário

Para que ocorra a transformação do modelo *weaving* para o PSM abstrato, é necessário que haja o modelo intermediário que visa oferecer suporte a criação de classes, propriedade e métodos semelhantes a um metamodelo UML [69]. O metamodelo Intermediário estabelecerá um vínculo entre os demais metamodelos, possibilitando a criação de nós que possam obter características dos demais modelos (PIM e PDMs).

A Figura 29 mostra o metamodelo intermediário proposto baseado em Matos [69], teve adaptações em suas classes para que fosse possível criar relacionamentos com mais de um metamodelo. O metamodelo Intermediário é composto por:

- **Merge:** define os elementos a serem utilizados do modelo da esquerda PIM e dos modelos da direita PSMs; funciona como uma fusão dos elementos utilizados, entrelaçando as classes dos elementos;
- **MergeElements** elemento que armazena as informações necessárias dos modelos originais da fusão;
- **Class:** reflete as classes provenientes dos modelos de entrada como os PDMs e PIM. Contém as definições de prioridade e operações que são fornecidas através das classes dos modelos de entrada.

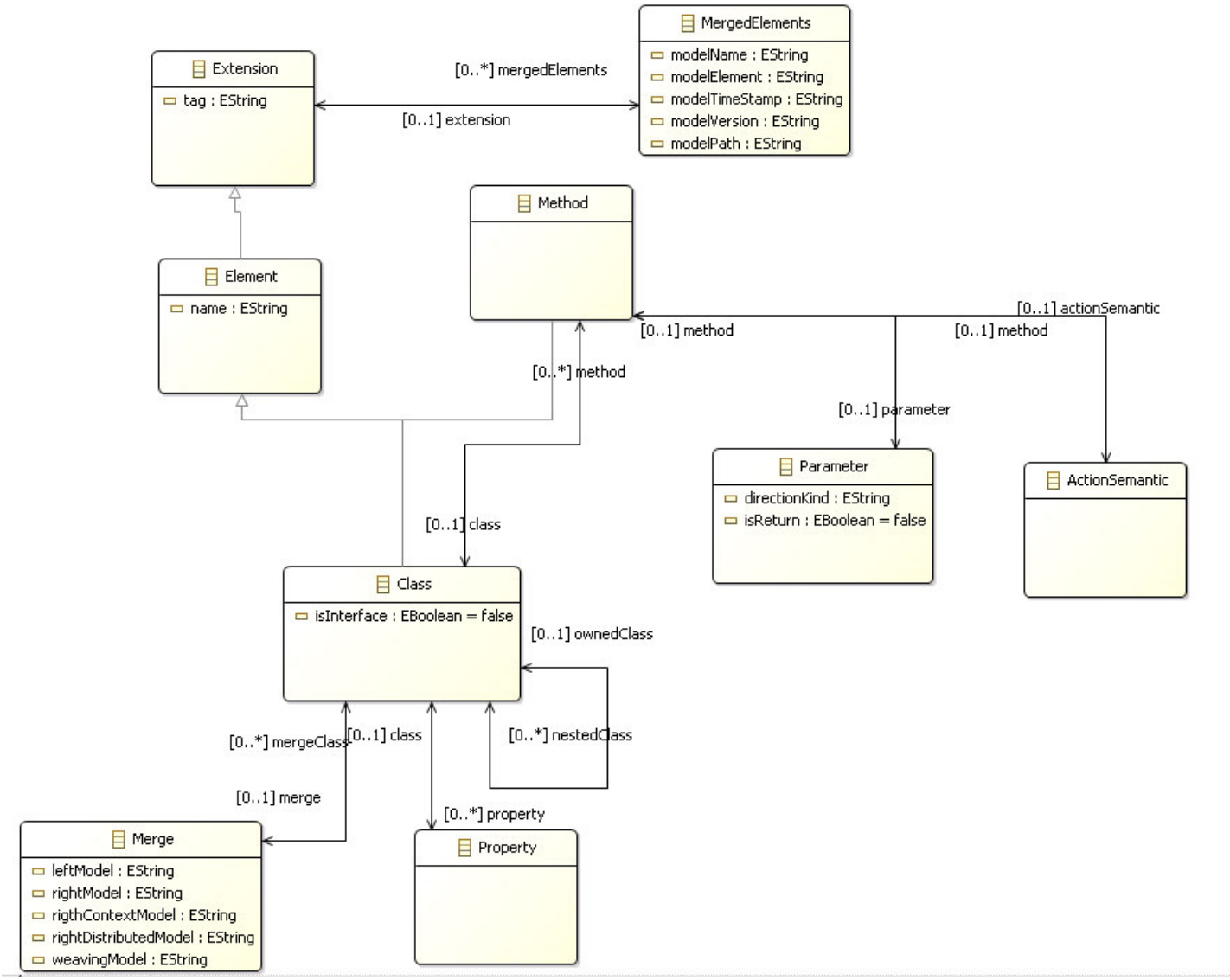

Figura 29: Metamodelo Intermediário baseado em [69]

4.3.6 Metamodelo para PSMs

O *framework* proposto apresenta em seu escopo os PSMs Abstrato e Concreto, sendo que cada um deles está em conformidade com seu respectivo metamodelo.

Para o *Abstract* PSM utilizamos o metamodelo de *web service* apresentado na Figura 30, que utiliza os padrões da linguagem WSDL e é representada por meio dos elementos:

- *Definition*: elemento principal de todo documento WSDL contendo atributos que definem os *namespaces* utilizados no documento WSDL [69];
- *Service*: identifica e define a localização real dos serviços;
- *Binding*: mapeia os elementos de operação em um elemento *PortType* para um protocolo específico;
- *Message*: o elemento *Message* corresponde as operações do *Web Service* definindo os dados a serem transmitidos, e se relaciona com os elementos Part que formam as partes reais da operação [69].

O metamodelo Java pode ser visto na Figura 31 e tem como objetivo ser o Metamodelo base para a criação do Concrete PSM, seus principais elementos são:

- JPackage: contêiner que possui todas as classes e interfaces de Java;
- JClassifier: base para os elementos *JPrimitiveType*, *JClass* e *JInterface*;
- JClassifier: define as associações de herança e contém o elemento *JMember*, que é a base para os elementos *JField* e *JMethod*;
- JClass: é uma especialização de um *JClassifier*, que contém os elementos *JField* e *JMethod* como membros;
- JField: composição do *JValue*, e tem um tipo *JPrimitiveType*, *JClass* ou *JInterface*, através do *JClassifier*;
- JMethod: possui a assinatura da operação, o elemento JParameter e o corpo do método;

- JParameter: corresponde ao argumento de um *JMethod*, podendo ser de entrada ou de saída.

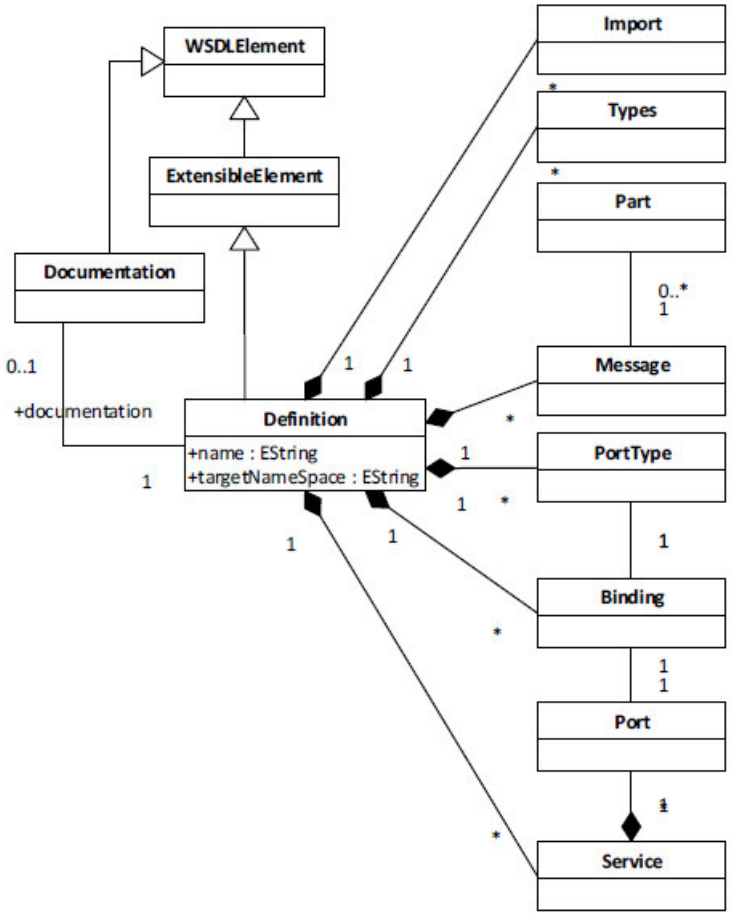

Figura 30: Metamodelo *Web Services* baseado em [69] e [45]

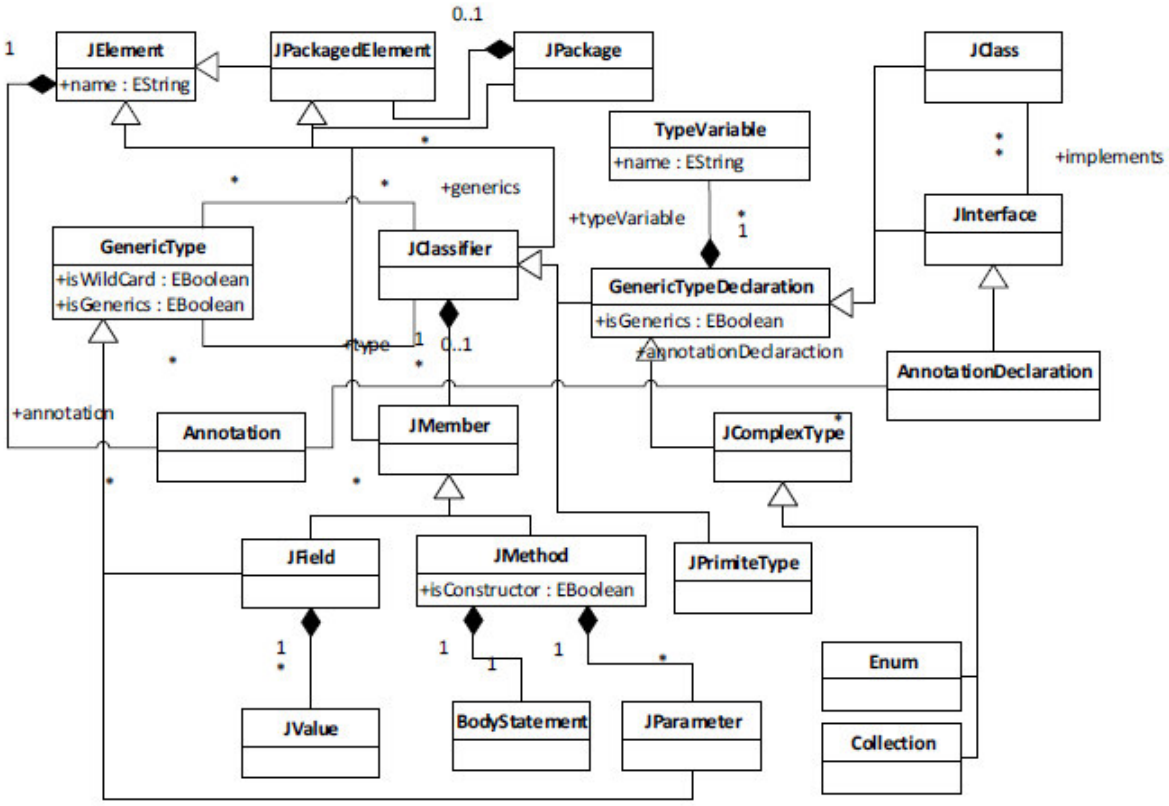

Figura 31: Metamodelo *JAVA* [69] e [45]

4.4 Síntese

Neste capítulo, um *framework* baseado em MDE e *Weaving* para suportar o desenvolvimento de sistemas sensíveis ao contexto foi apresentado. Embora este *framework* possa ser estendido para conter outros PDMs, nós nos focamos em apresentar PDMs para segurança, distribuição e sistemas sensíveis ao contexto.

Os metamodelos que fazem parte do framework prposto foram apresentados, suas características e definições, assim como, a metodologia para aplicação do *framework* proposto.

5 PROTOTIPAGEM E EXEMPLO ILUSTRATIVO

Neste capítulo, um protótipo é apresentado, demonstrando a implementação do *framework* proposto para suportar o desenvolvimento de sistemas de software sensíveis ao contexto. Logo em seguida, um exemplo ilustrativo será utilizado para demonstrar a utilização deste protótipo.

Este exemplo consiste na utilização do modelo PIM, cuja lógica de negócios consiste em obter os dados referentes a bateria, cpu e memória de um dispositivo, para que então possa ser realizado um compartilhamento de recursos entre dispositivos móveis que estejam conectados a uma rede de infraestrutura. A lógica de negócio consiste basicamente em obter apenas os dados necessários do *hardware e software* do dispositivo, dados como: cpu, memória e bateria.

5.1 Implementação do protótipo do framework proposto

Para implementar o *framework* proposto foram desenvolvidas as seguintes partes do protótipo:

- Editor *Weaving* versão 0.1.1 e *Weaving Tool* versão 0.1.1;
- Editor PDM baseado no metamodelo apresentado na Figura 25 e 26;
- Editor PDM baseado no metamodelo *context aware* apresentado na Figura 27;
- Editor PDM baseado no metamodelo de Sistemas Distribuídos apresentado na Figura 28;
- Gerador de modelo intermediário baseado no metamodelo apresentado;
- Definições de transformação apresentadas na Listagem.

Este protótipo foi desenvolvido usando as seguintes ferramentas:

- Java SE Development Kit (J2SDK) versão 7 [84];
- IDE Eclipse versão Mars Service Release 1 [85];

- Eclipse Modeling Framework versão 2.6.0 [88];
- ATL SDK para Eclipse versão 3.5.0 [86];
- Editor UML Papyrus versão 0.9.2 [87].

Utilizando a abordagem MDE e a técnica de *Weaving* criou-se uma ferramenta capaz de auxiliar na condução do *framework*. Essa ferramenta tem como premissa a capacidade de entrelaçar em paralelo distintos modelos. Esse entrelaçamento é realizado pela técnica de *weaving* e o gerenciamento dessa atividade é feito utilizando a MDE.

A ferramenta foi construída com base no trabalho de MATOS [69]. Ferramenta construída em Java e sendo passível a expansão.

A Figura 32, Figura 33 e Figura 34 mostram os códigos responsáveis pelo desenvolvimento da ferramenta, as Figuras ressaltam os código provenientes por percorrer a árvore do respectivo modelo e criar a partir do modelo percorrido sua respectiva árvore. Os *ModelHandler* são responsáveis por navegar entre os modelos dispostos na ferramenta e assim instanciar os elementos que farão parte do novo modelo.

```
//LISTAR OS ELEMENTOS DO MODELO DA ESQUERDA
    private EList getHandlersLeft(Namespace namespace, ModelHandler leftmodelhandler) {
        BasicEList list = new BasicEList();
        if (!(namespace.getOwnedElement().isEmpty())) {

            //NAVEGAR NO NAMESPACE E VERIFICAR SEUS PACOTES
            for (Iterator<umlmetamodel.ModelElement> nIter = namespace.getOwnedElement().iterator(); nIter.hasNext();) {
                umlmetamodel.Package pck = (umlmetamodel.Package) nIter.next();
                if(pck instanceof umlmetamodel.Package){
                    umlmetamodel.Package umlPackage = (umlmetamodel.Package) pck;
                    ElementModelHandler packageHandler = weavingFactory.createElementModelHandler();
                    packageHandler.setName(umlPackage.getName());
                    packageHandler.setPath(umlPackage.getName());
                    packageHandler.setMHandler(leftmodelhandler);
                    packageHandler.setType("Package");
                    list.add(packageHandler);

                    //NAVEGAR EM CADA PACOTE E VERIFICAR SUAS CLASSES
                    for (Iterator<umlmetamodel.ModelElement> cIter = pck.getOwnedElement().iterator(); cIter.hasNext();) {
                        umlmetamodel.ModelElement cla = (umlmetamodel.ModelElement) cIter.next();
                        if(cla instanceof umlmetamodel.Class){
                            umlmetamodel.Class umlClass = (umlmetamodel.Class) cla;
                            ElementModelHandler classHandler = weavingFactory.createElementModelHandler();
                            classHandler.setName(umlClass.getName());
                            classHandler.setPath(umlClass.getName());
                            classHandler.setType("Class");
                            classHandler.setHandler(packageHandler);

                            //NAVEGAR EM CADA CLASSE E VERIFICAR SEUS ATRIBUTOS E MÉTODOS
                            for (Iterator<umlmetamodel.ModelElement> classObjIter = ((umlmetamodel.Class) cla).getOwnedElement().i
                                umlmetamodel.ModelElement classObj = (umlmetamodel.ModelElement) classObjIter.next();
                                if(classObj instanceof umlmetamodel.Attribute){
                                    umlmetamodel.Attribute umlAttribute = (umlmetamodel.Attribute) classObj;
                                    ElementModelHandler attributeHandler = weavingFactory.createElementModelHandler();
                                    attributeHandler.setName(umlAttribute.getName());
                                    attributeHandler.setPath(umlAttribute.getName());
                                    attributeHandler.setType("Attribute");
                                    attributeHandler.setHandler(classHandler);
                                }
```

Figura 32: Código I - Ferramenta Weaving

```
//LISTAR OS ELEMENTOS DO MODELO DA DIREITA PDMCONTEXT
        private EList getHandlersRightContext(ContextAwareSystem context, ModelHandler rightmodelhandler1) {
            BasicEList list1= new BasicEList();
            if (!(context.getAdaptationcontract().isEmpty())) {

                for (Iterator roleIter = (context.getAdaptationcontract()).iterator(); roleIter.hasNext();) {

                    pdmmetamodelcon.AdaptationContract observa = (pdmmetamodelcon.AdaptationContract) roleIter.next();
                    ElementModelHandler roleHandler = weavingFactory.createElementModelHandler();

                    roleHandler.setName(observa.getName());
                    roleHandler.setPath(observa.getName());
                    roleHandler.setMHandler(rightmodelhandler1);
                    roleHandler.setType("Contexto");
                    list1.add(roleHandler);
```

Figura 33: Código II- Ferramenta Weaving

```
//LISTAR OS ELEMENTOS DO MODELO DA DIREITA1  (alterou apenas para o numero 1
private EList getHandlersRightDistributed(PortOwner distributed, ModelHandler rightmodelhandlerDistribuido) {
    BasicEList list1= new BasicEList();
    if (!(distributed.getPort().isEmpty())) {

        for (Iterator roleIter = distributed.getPort().iterator(); roleIter.hasNext();) {
            pDMMetamodelDistributed.Port role = (pDMMetamodelDistributed.Port) roleIter.next();
            ElementModelHandler roleHandler = weavingFactory.createElementModelHandler();
            roleHandler.setName(role.getName());
            roleHandler.setPath(role.getName());

            roleHandler.setMHandler(rightmodelhandlerDistribuido);
            roleHandler.setType("Distributed");
            list1.add(roleHandler);
```

Figura 34: Código III- Ferramenta Weaving

Após a codificação da ferramenta, *plug-ins* são gerados através de todos os metamodelos existentes no *framework*, para então poder executar a ferramenta na segunda instância do ECLIPSE, a exemplo da Figura 35 que representa os *plugins* gerados a partir dos metamodelos.

Para gerar os *plugins* foi necessário criar para cada modelo Ecore um genmodel e para cada gnmodel criar todos os *plugins*, essa criação é realizada através da própria ferramenta Eclipse, uma vez que os modelos estejam validados (Figura 35 e 36). Na Figura 36 retrata a criação dos plugins. De acordo com a Figura 36 ao clicar em *Generate All*, todos os plugins são gerados, sendo assim possível utiliza-los em uma segunda instância do Eclipse.

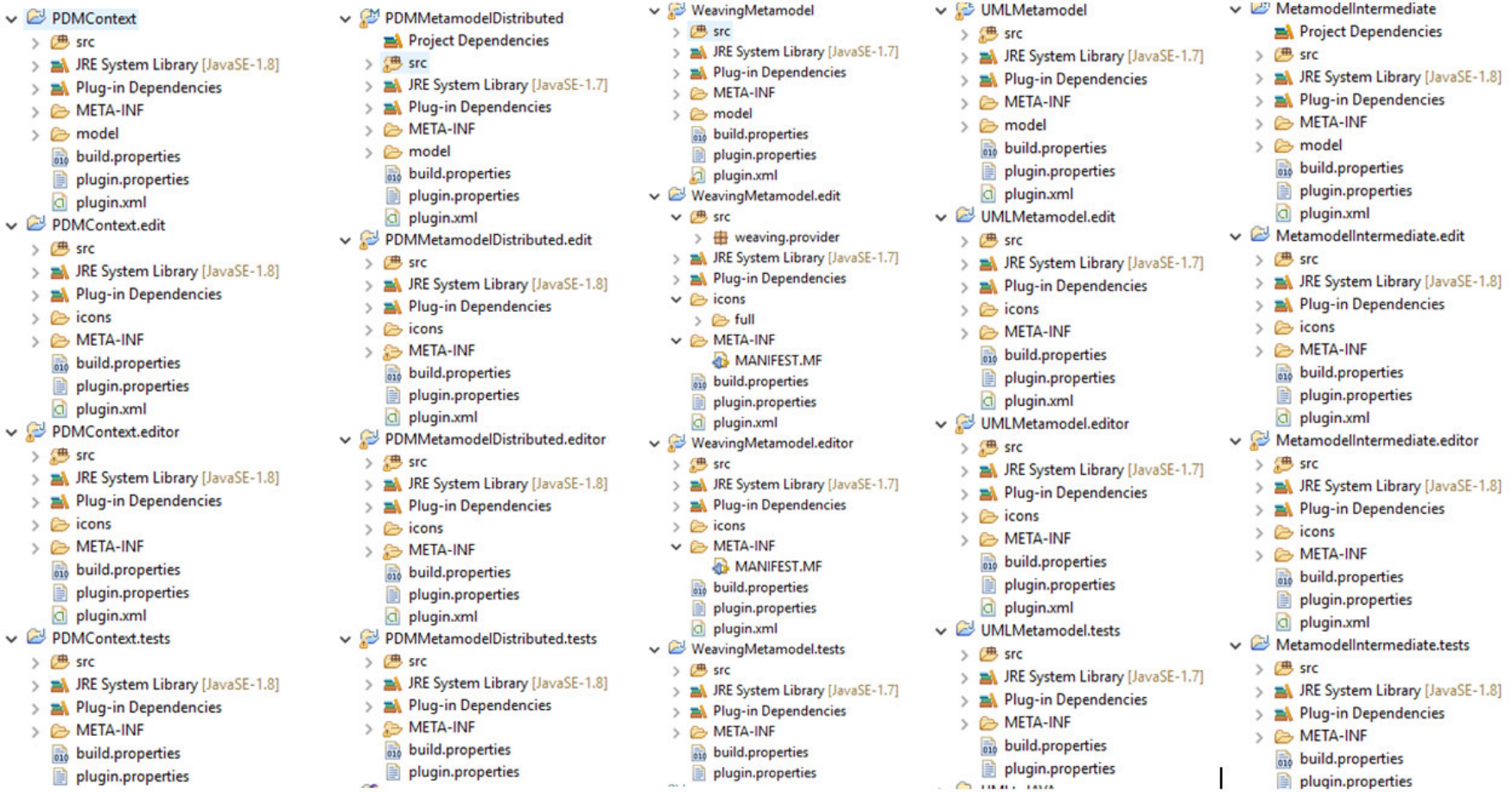

Figura 35: Protótipo: Plug-in para o IDE Eclipse

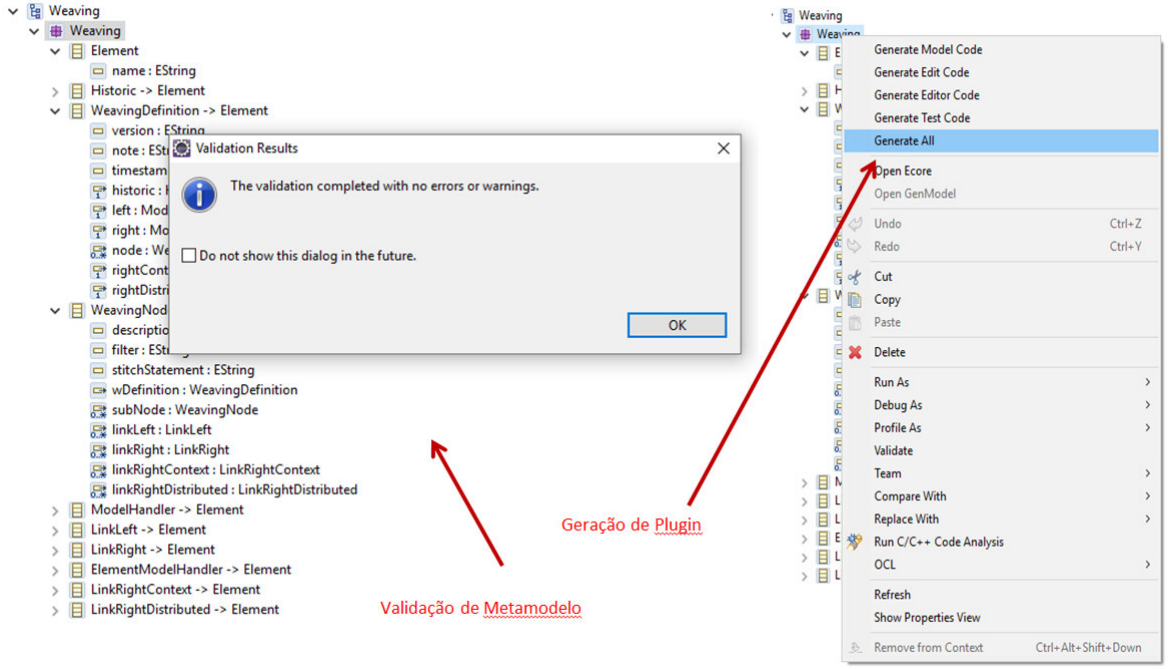

Figura 36: Geração de plug-in

Após a criação do modelo *weaving*, será criado o modelo intermediário com base no modelo de *weaving* e então passaremos para as transformações. Como exemplo temos a Figura 37.

Três definições de transformações foram utilizadas para as etapas do *framework*. A primeira realiza as definições de transformação do modelo intermediário para o *Abstract* PSM, a segunda realiza as transformações do *Abstract* PSM para o *Concrete* PSM e por último as definições de transformação do *Concrete* PSM para o*source code*. As regras ATL, visam transformar os elementos de um modelo em outros elementos de outro modelos já pré-determinado, a exemplo temos a Figura 37.

```
rule Class2Service {
	from class : INTERMEDIATE!Class
	to service : WSDL!Service (
		name <- class.name,
		ownerServ <- class.merge,
		port <- port
		),
		port : WSDL!Port (
			name <- class.name + 'Port',
			binding <- binding
		),
		binding : WSDL!Binding (
			name <- class.name + 'Binding',
			ownerBind <- class.merge,
			type <- portType
		),
		portType : WSDL!PortType (
			name <- class.name,
			ownerPTyp <- class.merge,
			binding <- binding
		)
}
```

Figura 37: Regra de transformação - ilustração

5.2 Exemplo ilustrativo

A ferramenta de *Weaving* foi criada com base no projeto de [69]. O exemplo irá retratar uma aplicação que deverá ser criada para obter dados de contexto de um *Smartphone* , dados como já mencionados: cpu, memória e bateria. Conforme os dados de contexto, a aplicação deverá ao obter uma atividade excolher entre processa-la de forma local ou remota. O esboço da codificação da aplicação irá ser gerada utilizando o *framework* aqui proposto. O modelo desta aplicação pode ser visto na Figura 38.

Após o PIM ter sido criado com base no metamodelo de UML, os PDMs devem ser criados de acordo com seus respectivos metamodelos. Neste exemplo ilustrativo, criamos o PDM de segurança, PDM de contexto e PDM de Sistemas Distribuídos, conforme pode ser visto na Figura 39.

Após a criação destes PDMs a ferramenta de *Weaving* é carregada manualmente com os PDMs e PIM. A Figura 40 mostra a *interface* que pede ao usuário que diga qual modelo deverá ser carregado, cada campo é preenchido com os modelos que devem já ter sido criados.

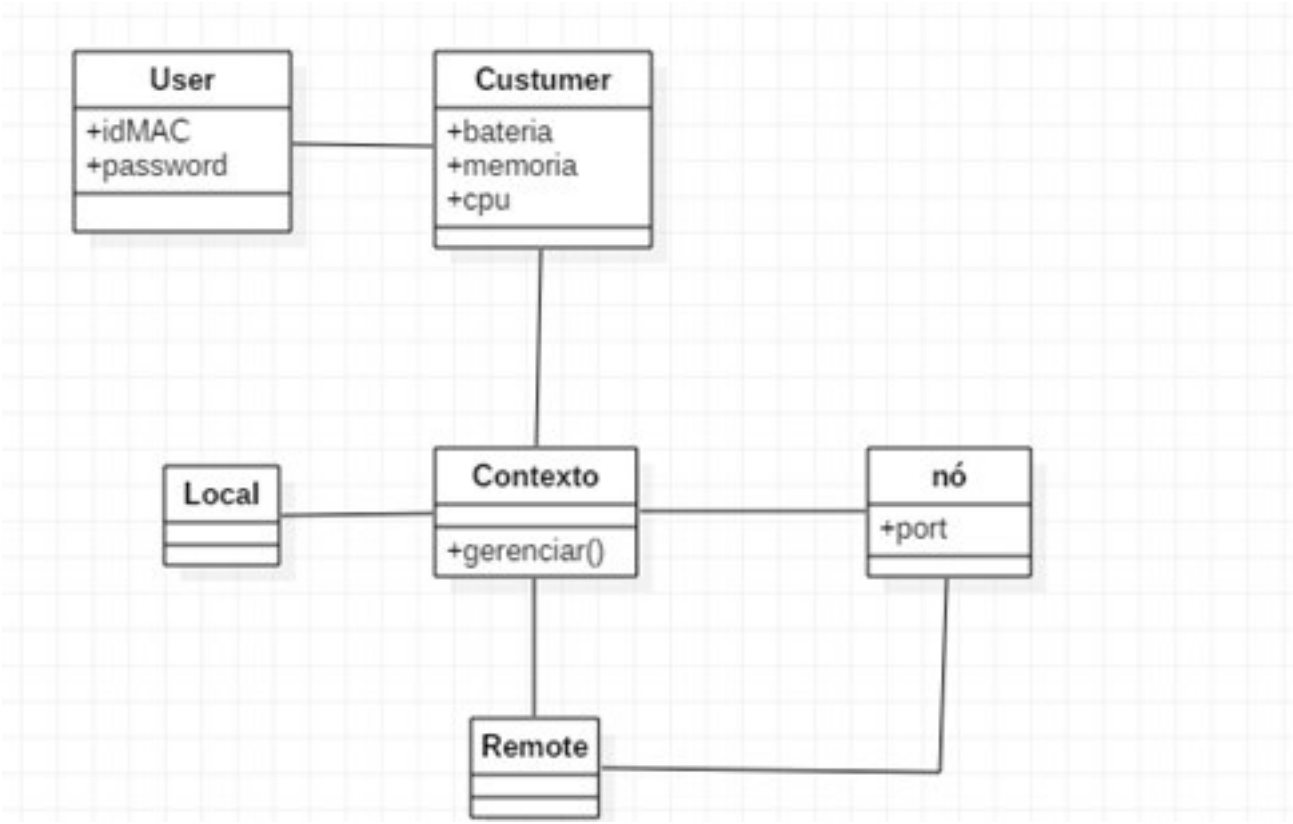

Figura 38: PIM - Lógica de negócio

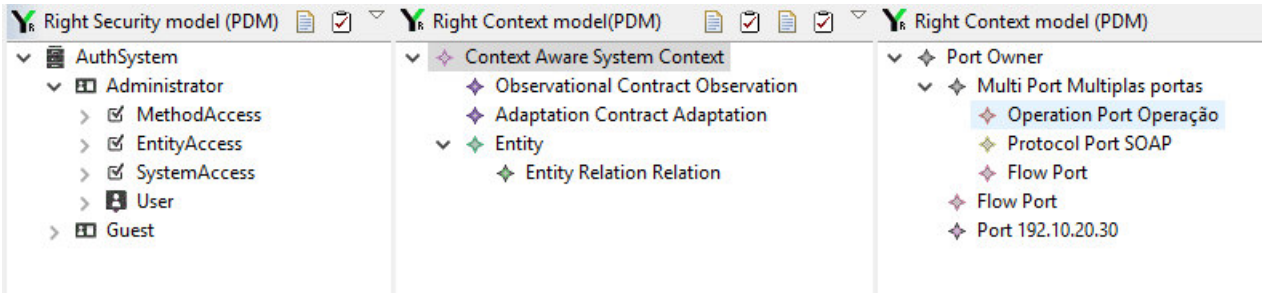

Figura 39: PDMs - PDM de Segurança, PDM de Contexto e PDM de Sistemas Distribuídos.

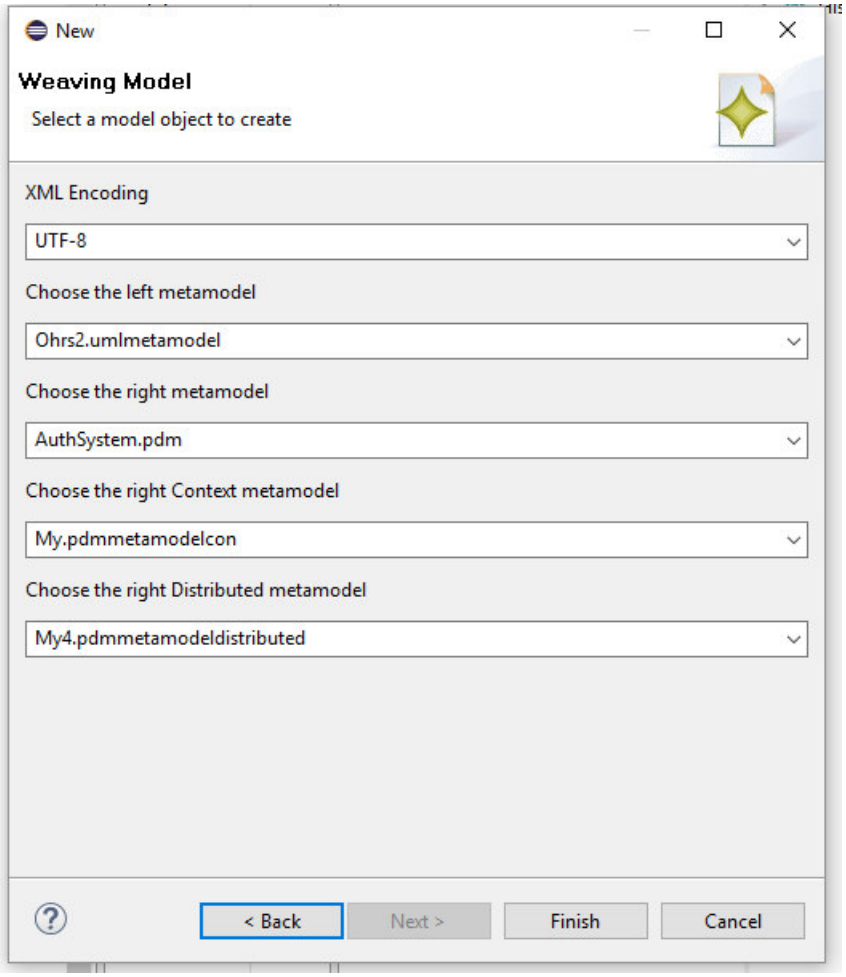

Figura 40: Carregamento de Modelos na ferramenta *Weaving*

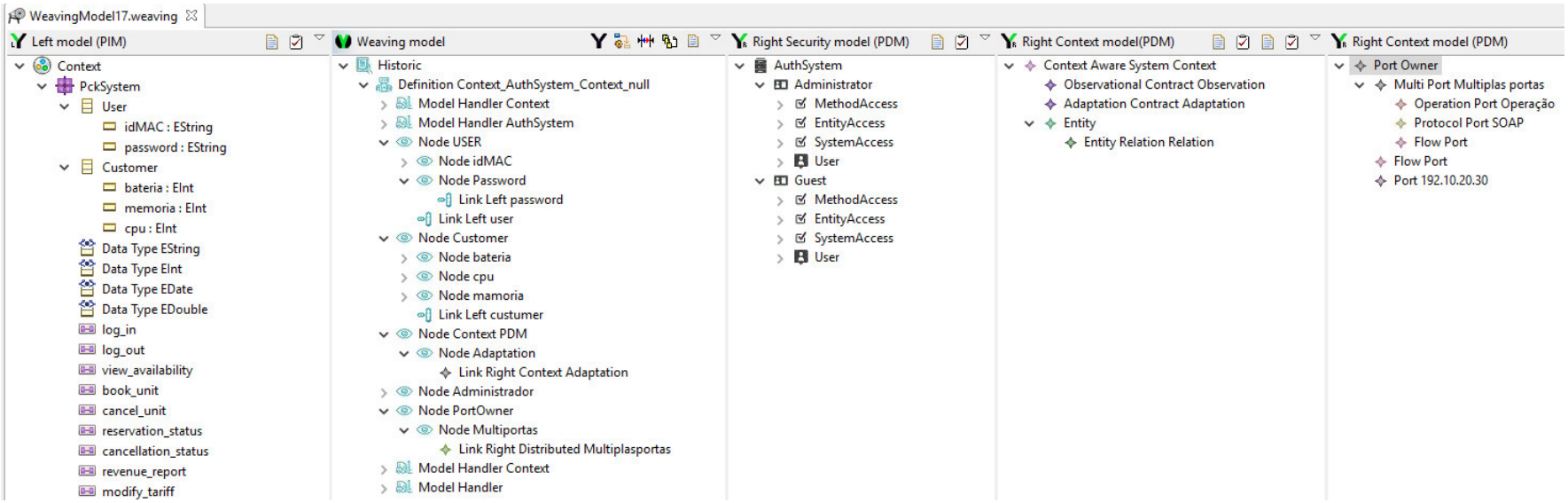

Figura 41: Plug-in para IDE-Eclipse: Entrelaçamento entre PIM, PDM para segurança, PDM para sistemas sensíveis ao contexto e PDM para distribuição.

Após esse carregamento os modelos são entrelaçados paralelamente gerando a *interface* vista na Figura 41. Essa ferramenta entrelaça os modelos de forma paralela, porém os nós de modelo para outro, é criado manualmente, conforme Figura 42. Assim, após este processo teremos então nosso modelo *Weaving*. Este modelo contém as informações do entrelaçamento do PDM de contexto, PDM de segurança e PDM de sistemas Distribuídos, assim como, possui os elementos do PIM.

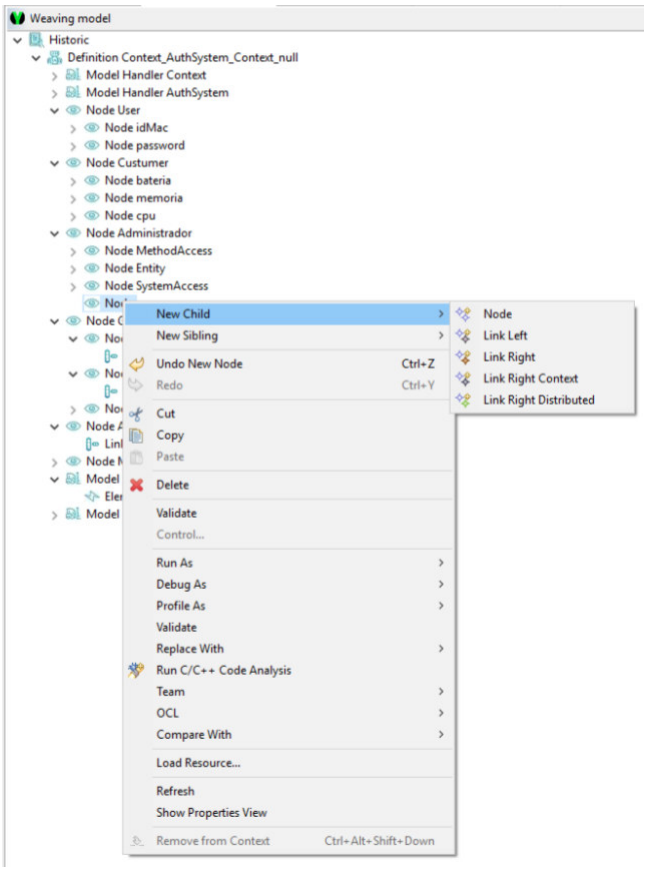

Figura 42: Criação dos nós do Weaving Model

A partir do modelo *Weaving*, criamos nosso modelo Intermediário, que contém as informações do entrelaçamento dos elementos de um PIM, de um modelo Weaving e de um PDM, mas que ainda possui elementos do modelo intermediário original. O modelo intermediário foi criado a partir do metamodelo Intermediário e geração de *plugins*, sendo que, seus elementos foram criados manualmente. A Figura 29 retrata esse modelo.

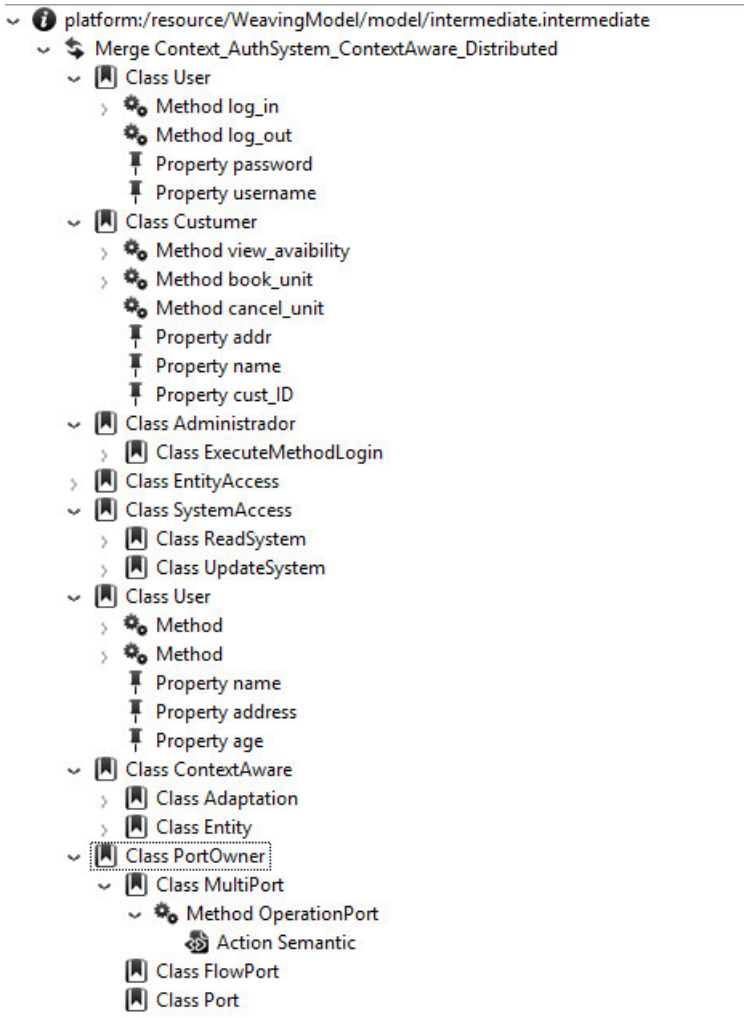

Figura 43: Modelo intermediário

Após obtermos o modelo Intermediário, regras de transformações são escritas e executadas. A primeira transformação resulta no *Abstract PSM* que deve estar em conformidade com o metamodelo de *WebService*, este servirá para auxiliar o sistema a ter mais capacidade de comunicação, caso necessário, via *web*. Outras definições de transformação são realizadas, chegando então ao *Concrete PSM*, este está em conformidade com o metamodelo de Java.

Na Figura 44 foi obtido através das regras de transformação, onde este teve correspondências do metamodelo de *Web services* e o próprio modelo intermediário. Posteriormente foram definidas regras de transformação para obter o PSM Concreto, este partiu do PSM Abstrato que foi dado como entrada. O PSM concreto pode ser visto na Figura 31.

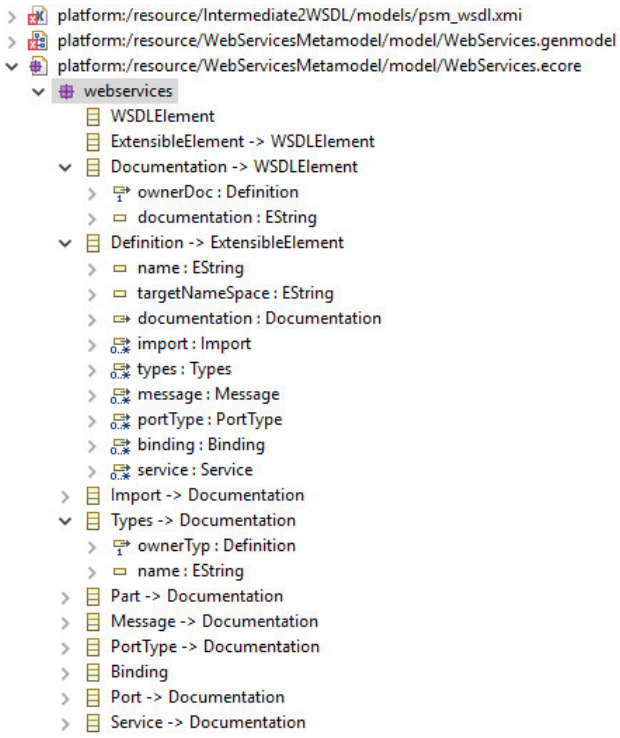

Figura 44: Abstract PSM

Por último, a geração de código. Na Figura 46 reflete uma das classes geradas a partir das transformações do PSM Concreto. Através das definições de transformação em ATL foi possível transformar modelo em texto com os elementos adquiridos desde

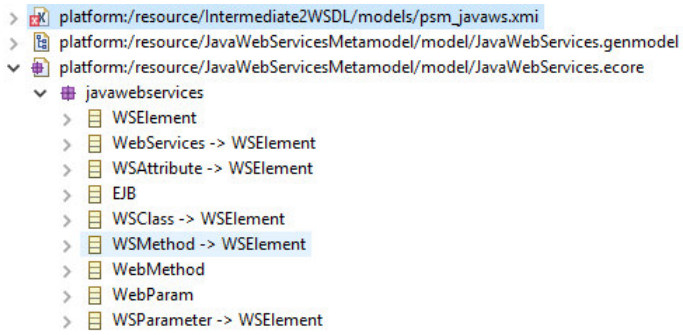

Figura 45: Concrete PSM

a criação do *Model Weaving*. É importante frisar que o código gerado é um esboço para a aplicação, que contém as principais características da aplicação, mas deverá ser customizada pelo desenvolvedor.

```
package ContextAwareSystem;

import javax.jws*

@ContextAwareSystem()
public class Entity {
  @EJB    public /*type*/ resource_id;
  @EJB    public /*type*/ resource_type;
  @EJB    public /*type*/ reference;

  @Method    public /*returnType*/ EntityRelationRelation() {    }
```

```
package AuthSystem;

import javax.jws.WebMethod;
import javax.jws.WebParam;
import javax.jws.WebService;

@WebService()
public class Administrator {
  @EJB    public /*type*/ resource_id;
  @EJB    public /*type*/ resource_type;
  @EJB    public /*type*/ reference;

  @WebMethod    public /*returnType*/ view_availability() {    }

  @WebMethod    public /*returnType*/ cancel_unit() {    }

  @WebMethod    public /*returnType*/ bateria_status() {    }

  @WebMethod    public /*returnType*/ cpu_status() {    }

  @WebMethod    public /*returnType*/ memoria_status() {    }
```

```
package PortOwner


@PortOwner()
public class MultiPort {
  @EJB    public /*type*/ resource_id;
  @EJB    public /*type*/ resource_type;


  @Method    public /*returnType*/ OperationPort() {    }

  @Method    public /*returnType*/ ProtocolPort() {    }

  @Method    public /*returnType*/ FlowPort() {    }
```

Figura 46: Trechos de Códigos-fonte

5.3 Análise dos Trabalhos Relacionados

Os seguintes critérios foram escolhidos como relevantes para a análise dos trabalhos apresentados neste capítulo:

- Tipo de abordagem utilizada no desenvolvimento de softwares;
- Mecanismo utilizado para unir as abstrações do sistema de software;
- Processo de desenvolvimento para aplicações sensíveis ao contexto;
- Permitir fazer correspondências entre elementos de bases/plataformas heterogêneas;
- Geração automática ou semi-automática de código-fonte;
- Reutilização de modelos;
- Entrelaçamento de plataformas heterogêneas;
- Suporte ao desenvolvimento de sistema de software

A Figura 47 retrata dentro os trabalhos pesquisados quais deles obedecem quais critérios previstos na Figura. Desta forma, analisando os trabalhos sobre a perspectiva de comparação, podemos considerar que:

Os trabalhos que utilizam em sua abordagem MDE para o desenvolvimento de *softwares*, como os de [42] e [41], apesar do foco de desenvolvimento não ser aplicações sensíveis ao contexto, demonstram a flexibilidade de se trabalhar com abstração. Todavia, estes trabalhos não suportam trabalhar paralelamente com plataformas heterogêneas.

Os trabalhos de [47] [76] e [77] são focados no desenvolvimento de sistemas sensíveis ao contexto. Esses trabalham utilizam diferentes abordagem em seu desenvolvimento, como SOA e *Systematic*, nenhuma delas permite a reutilização de modelos e não trabalha com o entrelaçamento ou fusão de plataformas heterogêneas que por sua vez não permite trabalhar plataformas heterogêneas paralelamente.

Os trabalhos que focam no desenvolvimento de sistemas sensíveis ao contexto com base na abordagem MDE, tais como os trabalhos de [78] e [75] não oferecerem suporte ao entrelaçamento de plataformas heterogêneas e por consequência não trabalha com a técnica de *Weaving*.

Nosso trabalho aborda todas os principais pontos analisados nesta pesquisa. Dando ênfase na capacidade de realizar o entrelaçamento paralelo de plataformas heterogêneas utilizando para isto a ténica de *Weaving* e a abordagem MDE. O trabalho também permite se reutilizar modelos e assim como os demais trabalhos fornece suporte ao desenvolvimento de sistema de *software*.

Elementos avaliados	**An approach based on MDE to support security in Softwareas a Service**	**A Framework Based on Model Driven Engineering to Support Schema Merging in Database Systems**	**A Framework for Developing Mobile, Context-aware Applications**	**A Middleware for Building Context-Aware Mobile Services**	**Mobile Gaia: A Middleware for Ad-hoc Pervasive Computing**	**A Model-Driven Approach to Generate Context Aware Applications**	**Model Driven Development of Context Aware Software Systems**	**UM FRAMEWORK PARA SUPORTE AO PROCESSO DE DESENVOLVIMENTO DE APLICAÇÕES SENSÍVEIS AO CONTEXTO**
Tipo de abordagem utilizada no desenvolvimento de softwares	**MDE**	**MDE**	**SYSTEMATIC**	**SOA**	**SOA**	**MDE**	**MDE**	**MDE**
Mecanismo utilizado para unir as abstrações do sistema de software	**Weaving**	**Não**	**Não**	**Não**	**Não**	**Não**	**Não**	**Weaving**
Processo de desenvolvimento para aplicações sensíveis ao contexto	Não	Não	Sim	Sim	Sim	Sim	Sim	Sim
Permitir fazer correspondencias entre elementos de bases/plataformas heterogêneas	Sim	Sim	Não	Não	Não	Não	Não	Sim
Geração automática ou semi-automática de código-fonte	Sim	Sim	Não	Não	Não	Sim	Sim	Sim
Reutilização de modelos	Sim	Sim	Não	Não	Não	Sim	Sim	Sim
Capacidade de trabalhar com plataformas heterogêneas em paralelo	Não	Não	Não	Não	Não	Não	Não	Sim
Suporte ao desenvolvimento de sistema de software	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim

Figura 47: Análise comparativa entre artigos

5.4 Síntese

Este capítulo apresentou um protótipo e um exemplo ilustrativo que demonstra sua utilização.

Apresentou-se um exemplo ilustrativo de uma aplicação que adquire dados de contexto de um *Smartphone*. Foi demonstrado passo a passo a utilização do *framework* aqui proposto.

Apresentou-se também a metodologia utilizada pelo *framework*, sendo correlatada a partir de um diagrama de atividades. Assim como, foram mostradas as gerações de plugins de acordo com seus metamodelos. Neste Capítulo também foi mostrado, o Modelo de *Weaving*, PDM de segurança, PDM de Contexto, PDM de Sistemas Distribuídos, ferramenta de *weaving*, modelo intermediário, PSM abstrato, PSM concreto e trechos de código fonte.

Também foi apresentada a análise comparativa entre os trabalhos presentes na literatura e o presente trabalho de Dissertação. Os critérios foram mostrados e discorrido sobre eles.

Os trabalhos apresentados foram: 2 (dois) trabalhos no item de abordagens baseada em MDE para desenvolvimento de softwares -An approach based on MDE to support security in Softwareas a Service e A Framework Based on Model Driven Engineering to Support Schema Merging in Database Systems, 3 (três) da área de Abordagens para o desenvolvimento de aplicações sensíveis ao contexto os quais citamos o trabalho de A Framework for Developing Mobile, Context-aware Applications, A Middleware for Building Context-Aware Mobile Services e Mobile Gaia: A Middleware for Ad-hoc Pervasive Computing e por último destacamos os artigos da área de MDE e aplicações sensíveis ao contexto: A Model-Driven Approach to Generate Context Aware Applications e Model Driven Development of Context Aware Software Systems.

É notório que várias pesquisas correlacionados ao desenvolvimento de sistemas de *software context aware* vêm sendo estudado pela comunidade científica, bem como, é

notório que as dificuldades de desenvolver estes tipos de aplicações ainda persiste para o desenvolvedores haptos a desenvolver estes tipos de sistemas.

6 CONCLUSÕES

Neste capítulo, iremos discutir os objetivos atingidos, as limitações do *framework* proposto e trabalhos futuros.

Este trabalho intitulado *Um Framework utilizando Model Driven Engineering e Weaving como suporte ao desenvolvimento de sistemas sensíveis ao contexto*, apresenta a proposta de um *framework* que se destaca por entrelaçar modelos de plataformas heterogêneos. Este *framework* auxilia no desenvolvimento dos sistemas por meio da MDE, além de trabalhar de forma abstrata utilizando modelos para isto.

Podemos considerar que a nossa estrutura portanto pode fornecer:

1. flexibilidade para refinar e adaptar novos elementos;
2. manter um registro da origem dos elementos na modelagem;
3. ter capacidade de trabalhar com a técnica de tecelagem para plataformas heterogêneas;
4. apoiar a geração de definições de transformação orientadas por um modelo de *weaving*.

6.1 Objetivos alcançados

Tendo em vista que nosso principal objetivo que é desenvolver um *framework* que dê suporte à aplicações sensíveis ao contexto, utilizando aspectos de segurança, contexto e sistemas distribuídos, além da técnica de *weaving*, os seguintes objetivos foram alcançados:

- Proposta de um *framework* que utiliza MDE e *Weaving* para fusão de plataformas heterogêneas, sendo estas plataformas: *context, security, distributed;*

- aplicação da técnica de *weaving* no processo de entrelaçamento de modelos que representam as aplicações;
- prover um processo de desenvolvimento que seja organizado por um padrão, onde nesta pesquisa utilizamos a especificação MDA;
- Prover uma ferramenta que entrelace modelos de plataformas heterogêneas, onde este entrelaçamento ocorra em uma única etapa, ao contrário do que a proposta de *weaving* prevê permitindo entrelaçar os modelos heterogêneos entretanto um a um em diferentes etapas.

6.2 Contribuição

Esta seção tende a apresentar as contribuições do trabalho de pesquisa no campo científico e tecnológico a partir do tema escolhido e descrito como se segue:

6.2.1 Científicas

A pesquisa apresentou as seguintes contribuições:

- *framework* de suporte ao desenvolvimento de sistemas sensíveis ao contexto: utilizou abordagem MDE e técnica de *Weaving* para melhoria no desenvolvimento destas aplicações;
- modelagem e propagação dos aspectos de *context aware* em vários níveis de modelagem, utilizando metamodelos e modelos;
- reutilização dos metamodelos de UML, *Weaving*, segurança, contexto e sistemas distribuídos adotados;
- utilização da técnica de *Weaving* e abordagem MDE para o entrelaçamento de plataformas heterogêneas em paralelo;

6.2.2 Tecnológicas

Se pode citar dentre as contribuições tecnológicas que :

- Desenvolvimento de um *framework* como plug-in para a ferramenta Eclipse, incluindo EMF e ATL;
- criação e adaptação dos metamodelos de *Weavin*, Segurança, Sistemas Distribuídos, Ciência de Contexto para geração dos seus respectivos modelos;
- criação de uma ferramenta que utiliza técnica de *Weaving* e MDE para entrelaçar distintos modelos de forma paralela;

6.3 Publicações

A presente pesquisa resultou em um artigo aceito na ICC'17 (*Second Internation conference on Internet of Things, Data and Cloud Computing*,que será Será realizada na cidade de Cambridge, Churchill College. University of Cambridge, United Kingdom. 22-23 March 2017. Todos os artigos aceitos serão publicados na ACM (*International Conference Proceeding Serie (ICPS* e está disponível na ACM Digital Library ISBN: 978-1-4503-4774-7

Título: *A Framework for supporting the context-aware mobile application development process*
Autores: Debora Stefanello e Dr. Denivaldo Lopes

6.4 Limitações e trabalhos futuros

O trabalho atinge nosso objetivo principal, porém os modelos de segurança, não se aproximam de conceitos de criptografia. A entrelaçamento dos modelos não ocorre automaticamente. Os modelos são instanciados manualmente através de EMF usam por padrão o editor em forma de árvore. A estrutura executa as transformações de modelos usando a linguagem de transformação ATL: eles poderiam ser usados outros idiomas como MOF QVT, MOFScript.

Para possíveis trabalhos futuros, a ideia seria automatizar o processo de entrelaçamento de modelos por meio da técnica de *weaving*, além de criar uma API para aplicações Móveis, onde o *framework* possa utilizar essa API para complementar os códigos fornecidos para a aplicação.

Os modelos utilizados são instanciados manualmente e são exibidos em forma de árvore utilizando o EMF. A utilização em forma gráfica poderia deixar a ferramenta desenvolvida mais intuitiva.

Ampliar as atribuições do código-fonte com mais recursos do metamodelo de sistemas distribuídos, contexto e segurança.

Referências

[1] GRAMA A. G . V. K, KARYPIS G.. **Introduction to Parallel Computing: Design and Analysis of Algorithms.** Addison-Wesley. 2008.

[2] PITANGA M. **Construindo Supercomputadores com Linux.** BRASPORT. ISBN 9788574523729 ,2008.

[3] GRIEBE T, GRUHN V. **A Model-Based Approch to Test Automation for Context-Aware Mobile Application**.Proceedings of the 29th Annual ACM Symposium on Applied Computing. 2014

[4] ALBRECHT S., KOFI A. A., ANTTI T., URPO T.,KRISTOF V. L., and WALTER V. de V. **Advanced Interaction in Context in Handheld and Ubiquitous** Computing Springer-Verlag. 1999.

[5] ADDLESEE M, CURWEN R., HODGES S., NEWMAN J., STEGGLES P.,WARD A.,HOPPER A. **Implementing a Sentient Computing System** IEEE Computer Magazine, Vol. 34. 2001.

[6] LOPEZ D. de I., MENDONA P. and HOOPER A. ~ **TRIP: a Low-Cost Vision-Based Location System for Ubiquitous Computing**. Personal and Ubiquitous Computing journal, Springer. 2002.

[7] FABRO D., BÉZIVIN J., JOUALT F., and VALDERIEZ P. **Applying Generic Model Management to Data Mapping**. Proc. of Base de Données Avancées Saint-Malo, France. 2005.

[8] SATO K. **Context Sensitive Interactive Systems Design: A Framework for Representation of contexts**. Proceedings of the HCII 2003 Conference, Vol. 3, Lawrence Erlbaum Associates, Mahwah, New Jersey. 2003.

[9] SATO K. **Context-Sensitive Design; Bridging Viewpoints for Human-Centered Design**, Proceedings of the FutureGround 2004 Conference, Design Research Society, Melbourne 2004.

[10] HONG J. SUH E. KIM S. **Context-aware systems: A literature review and classification**. Jornal Expert Systems with Applications. 2009.

[11] SANTOS T.T.,PAVÃO R.K. , QUADRA A.S. e LEMOS F.A.B. **Aquisição de Dados Remotos Provenientes de Pequenas Centrais Hidroelétricas via Telefonia Celular**. IEEE, São Paulo [em CD]. (2004)

[12] SOMMERVILLE, I. **Software Engineering**, 8st ed. Pearson, 2007.

[13] SCHILIT B., THEIMER M. . **Disseminating Active Map Information to Mobile Hosts**. IEEE Network. 1994.

[14] HENRICKSEN K. INDULSKA J. McFADDEN T. BALASUBRAMANIAM S. **Middleware for Distributed Context-Aware Systems**. Springer - OTM Confederated International Conferences "On the Move to Meaningful Internet Systems". 2005.

[15] SERRAL E. VALDERAS P. PELECHANO V. **Towards the Model Driven Development of context-aware pervasive systems** . Jornal Pervasive and Mobile Computing. Elsevier. 2009.

[16] MOSTÉFAOUI G. K., ROCHA J. P., and BRÉZILLION P. **Context-Aware Computing: A Guide for the Pervasive Computing Community.** In Proceedings of The IEEE/ACS International Conference on Pervasive Services (ICPS'04), Beirut, Líbano. 2004

[17] CHEN G.,KOTZ D. **A Survey of Context-Aware Mobile Computing Research**. Technical Report TR2000-381, Dartmouth College, Computer Science, Hanover, EUA, 2000.

[18] FRANCE R., RUMPE B. **"Model-driven development of complex software: A research roadmap"**. 29th International Conference on Software Engineering: IEEE Computer Society. 2007.

[19] SCHMIDT, D. C. **Guest Editor's Introduction: Model Driven Engineering**. IEEE Computer Society. 2006.

[20] HAFIDDI H., BAIDOURI H., NASSAR M., EL ASRI B., HRIOUILE A. **"Context-Aware Service Centric Approach for Service Oriented Architectures**. 13th International Conference on Enterprise Information Systems (ICEIS'11).2011.

[21] SALVIATO T., COSTA P., GONÇALVEZ J. VALE I. **Framework for Context-aware Applications on the Brazilian Digital TV**. Fourth International Conference on Ubi-Media Computing. 2011.

[22] DEY, A. K. *Understanding and using context*. Personal and Ubiquitous Computing. 2001

[23] KNOW, O. B. **Modeling and generating context-aware agent-based applications with amended colored petri nets.** Expert Systems with Applications. 2004.

[24] LUCRÉDIO, D. **"Uma Abordagem Orientada a Modelos para Reutilização de Software"**. Tese de Doutorado, Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, SP, 2009.

[25] ZENDAGUI, B. **A model-driven engineering approach for the observation needs specification**. 2009

[26] BRÉZILLON, P. **Context in problem solving: a survey, The knkowledge Engineering Review**. 1999.

[27] DEY, A.K.; ABOWD , GREGORY D.**Towards a Better Understanding of Context and Context-Awareness**. 1st International Symposium on Handheld and Ubiquitous Computing, 1999.

[28] DEY, A.K.; SALBER, D.; ABOWD, G.D.**A conceptual framework and tookit for supporting the rapid prototyping of contexto-aware applications**. HumanComputer Interaction Journal. 2001.

[29] STEINGERG D., BUDINSKY F., PATERNOSTRO M. and MERKS E. **EMF: Eclipse Modeling Framework**, 2nd ed. Addison-Wesley Professional. 2008.

[30] KLEPPE, A., WARMER, J., and BAST, W.**MDA Explained: The Model Driven Architecture: Practice and Promise**, 1st ed. Addison- Wesley. 2003

[31] OMG. **MDA Guide Version 1.0.1**, Document Number: omg/2015-06-12. OMG, June 2015.

[32] SANTOS, V. V. **CEManTIKA: A domain independente framework fo designing contexto-sensitive systems**. Tese de Doutorado. Universidade Federal de Pernambuco. Centro de Informática, Recife, PE. 2008.

[33] MERRIAN-WEBSTER. Disponível em : www.merriam-webster.com . Acessado em 26 de out de 2016.

[34] CHEN, G.; KOTZ, D. **A Survey of Context-Aware Mobile Computing**. Dartmouth Computer Science Technical Report TR2000-381. 2000.

[35] CARVALHO, M. V. R. **"Uma abordagem baseada na engenharia dirigida por modelos para suportar merging de base de dados heterogêneas"**. Dissertação de Mestrado, Departamento de Engenharia Elétrica, Universidade Federal do Maranhão. 2014.

[36] LOPES, D.; ABDELOUAHAB, Z. **ESOW: Parallel/Distributed Programming on the Web**. 13th International Workshop on Database and Expert Systems Applications. 2002.

[37] BÉZIVIN J. ,HAMMOUDI S. , LOPES D. ,and JOUAUT F. **B2B Applications, BPEL4WS, Web Services and .NET in the Context of MDA, Knowledge** 107 Sharing in the Integrated Enterprise, Springer. 2005.

[38] GROUP, A., LINA, AND INRIA. **Atlas Transformation Language - ATL user Manual version 0.7**. Nantes. 2008

[39] LOPES D. **Étude et applications de I'approache MDA por des plates-formes de Services Web**. Ph.D Thesis, Université de Nates. 2005.

[40] CARVALHO, M. V., ABDELOUHAB, Z., LOPES D., **Uma Abordagem baseada em Engenharia Dirigida por Modelos para suportar Merging**

de Base de Dados Heterogêneas, Dissertação de Mestrado - Universidade Federal do Maranhão. 2014.

[41] CARVALHO, M. V., ABDELOUHAB, Z., LOPES D. **A Framework Based on Model Driven Engineering to Support Schema Merging in Database Systems**.Lecture Notes in Electrical Engineering. 2014.

[42] MATOS P. LOPES D. ABDELOUAHAB Z. **A Model Driven Engineering Approach to Support the Development of Secure Software as a Service** Journal of Software. 2016.

[43] LOPES, D., HAMMOUDI, S., AND ABDELOUAHAB, Z. **Schema Matching in the Context of Model Driven Engineering: From Theory to Practice.**Proceedings of the International Conference on Systems, Computing Sciences and Software Engineering (SCSS 2005). 2005.

[44] OASIS. **Reference architecture for service architecture version 1.0**. 2008

[45] BÉZIVIN J.,HAMMOUDI S. , LOPES D., JOUAULT F. **Applying MDA Approach for Web Service Platform**. Proc. of the 8th Enterprise Distributed Object Computing Conference (EDOC 2004) – IEEE, 2004.

[46] DOKOVISK, N. **Paradigm: Service oriented computing**. University of Twente. 2004.

[47] GREGORY B., CAHILL V. **A Framework for Developing Mobile, Context-aware Applications**. 2nd IEEE Conference on Pervasive Computing and Communications. 2014.

[48] GWIZDKA J. **What's in the context. In Proceedings of the Computer Human Interaction 2000 (CHI2000)**, Workshop on "The What, Who, Where, When, Why and How of Context-Awareness", Hague, Holanda, 2000.

[49] AMORIM, S. **A tecnologia web service e sua aplicação num sistema de gerência de telecominicações.** Dissertação de Mestrado. UNICAMP. 2004.

[50] JAVA. **Plataforma java**. Disposnível em: http://www.oracle.com/tecnetwork/java/. Acessado em 15/03/2016.

[51] W3C. **extensible markup language**. Disponível em: http://www.w3c.org/standards/xml/. Acessado em 15/03/2016.

[52] UML. **Unified modeling language**. Disponível em: http://www.omg.org/spec/spec/UML/. Acessado em 15/03/2016.

[53] STEINBERG, D.,BUDINSKY, F., PATERNOSTRO, M., and MERKS, E. **EMF:Eclipse Modeling Framework**, 2rd ed. Addinson-Weley Professional, 2008.

[54] EMF. Eclipse modeling framework project, 2013. **Eclipse Modeling Framework Project (EMF)**. Disponível em http://www.eclipse.org/modeling/emf/. Acessado em 15/03/2016.

[55] OMG.OMG. **Object Management Group**. Disponível em: http://omg.org/. Acessado em 15/03/2016.

[56] VIEIRA, V., TEDESCO, P., SALGADO, A. C. **Designing context-sensitive systems: An integrated approach.**Expert Systems with Applications, v. 38, n. 2, p. 1119-1138, 2011.

[57] SANTOS D. V.V **CEManTIKA: A domain-independent framework for designing context-sensitive systems**. Tese de Doutorado - Universidade Federal de Pernambuco, 2008.

[58] SILVA, V. G. **Descoberta Dinâmica, Sensível ao Contexto, de Serviços Web**. Dissertação de Mestrado - Universidade de São Carlos. 2014.

[59] ERL, T. **Service-orinted architecture - a field guide to integrating xml and web services.** 1st. ed. Prentice Hall, 2004.

[60] MITRA, N., LAFON, Y. SOAP version 1.2 Part 0: Primer (Second Edtion). Disponível em : http://www.w3c.org/TR/2007/REC-soap12-part0020070427/.

[61] PANTASSO, C., ZIMMERMANN, O.,LEYMANN, F. **RESTful web service vs. "Big"web services: Marking the rigth architectural decision**. In: Proceeding of the 17th International Conference on World Wide Web 2008, WWW'08.[SL.:s.n], p.805 - 814.

[62] SCOT S. **SOAP: cross plataform web service development using xml**. Prentice Hall, 2002.

[63] OLIVEIRA R.R. **Avaliação de manutenabilidade entre as abordagens de web services RESTful e SOAP-WSDL**.Dissertação de Mestrado. USP - São Carlos, 2012.

[64] MIGNON A. S. **Aplicação da Técnica de Tecelagem de Modelos na Transformação de Modelos na MDA**.Dissertação de Mestrado. USP - São Paulo, 2007.

[65] FABRO, M. D. D., BÉZIVIN J. JOUAULT F. BRETON E., GUELTAS G. **AMW: A Generic Model Weaver.**Proceedings of the 1ére Journée sur L'Ingénierie Dirigée par Modèles (IDM05). Paris[s.n], 2005.

[66] JOUAULT F., BÉZIVIN J. **KM3: a DSL for Metamodel Specification**. Proceeedings of 8th IFIP International Conference on Formal Method for Open Object-Based Disitributed Systems. LNCS 4037. Bologna, Italy. 2006.

[67] NUNES P. R. A. F.**Validação de padrões de web services transacionais**. Dissertação de Mestrado. USP - São Paulo. 2011.

[68] ABOWD G. D., MYNATT E. D. **Charting past, present, and future research in ubiquitous computing**. ACM Transactions on Computer-Human Interaction (TOCHI). 2000.

[69] MATOS P. L., **Um Framework de Segurança baseado em Engenharia Dirigida por Modelos para Plataformas de Computação em Nuvem: uma Abordagem para Modelos SaaS**. Dissertação de Mestrado. UFMA. 2015.

[70] LAFI, L., FEKI J., AND HAMMOUDI S. **Metamodel matching techniques evaluation and benchmarking**. International Conference on Computer Applications Technology. 2013.

[71] OMG. **Documents Associated With UML Profile For Enterprise Distributed Object Computing**, Version 1.0. em: http://www.omg.org/spec/EDOC/1.0/. Acess: 27-06-2016

[72] TRUONG K. N., ABOWD G. D., BROTHERTON J. A. **Who, What, When, Where, How: Design issues of capture & access applications**. In Proceeding of the International Conference on Ubiquitous Computing, pages 209-224.

[73] LOPES D., HAMMOUDI S., BÉZIVIN J., JOUAULT F. **Mapping Specification in MDA: From Theory to Practice**. First International Conference INTEROPESA '2005 Interoperability of Enterprise Software and Applications. 2005.

[74] ECLIPSE. ATL: Atlas Transformation Language. ATL Installation Guide. Version 0.1 2005

[75] SINDICO A., GRASSI V. **Model Driven Development of Context Aware Software Systems** . International Workshop on Context-Oriented Programming. 2009.

[76] GU T., PUNG H., ZHANG D. **A middleware for building context-aware mobile services** . Vehicular Technology Conference, 2004. VTC 2004-Spring. 2004.

[77] SHIVA C., MUHTADI H., CAMPBELL R., MICKUNAS D. **Mobile Gaia: A Middleware for Ad-hoc Pervasive Computing IEEE Consumer Communications**. Networking Conference, CCNC 2005.

[78] DUARTE P., BARRETO F., GOMES F., CARVALHO W., TRINTA F. **A Model-driven to Generate Context Aware Applications**. WebMedia.2014

[79] ALALFI M. H. , CORDY J. R. , and DEAN T. R. **Automated Verification of Rolebased Access Control Security Models Recovered from Dynamic Web Applications**. 14th IEEE International Symposium on Web Systems Evolution (WSE), Trento. 2012.

[80] LOPES D., HAMMOUDI S., BÉZIVIN J., JOUALT F. **Generating Transformation Definition from Mapping Specification: Application to Web Service Platform.** Lecture Notes in Computer Scince. 2005.

[81] VIEIRA V., TEDESCO P., SALGADO A. C. **Modelos e Processos para o Desenvolvimento de Sistemas Sensíveis ao Contexto**. UFBA. 2009.

[82] TACONET C., ZAZIA K., MEHDI Z., CONAN D. **CA3M: A Runtime Model and a Middleware for Dynamic Context Management**. Springer-Verlag Berlin Heidelberg.2009.

[83] BASIN D. ,DOSER J. , and LODDERSTEDT T. **Model Driven Security: from UML Models to Access Control Infrastructures**. ACM Transactions on Software Engineering and Methodology. 2006.

[84] W3C. XML Technology. Disponível em: http://www.w3c.org/standards/xml. Acesso em: 14/06/2016

[85] ECLIPSE, 2016. IDE Eclipse Mars. Disponível em: https://eclipse.org/mars/. Acesso em: 16/11/2016

[86] Eclipse, 2016. ATL – a model transformation technology. Disponível em: http://www.eclipse.org/atl/. Acesso em: 15/11/2016.

[87] ECLIPSE, 2016. Eclipse Projects – Papyrus. Disponível em: http://www.eclipse.org/papyrus/. Acesso em: 15/06/2016.

[88] Eclipse Modeling Framework (EMF). Eclipse Modeling Project. Disponível em: http://www.eclipse.org/modeling/emf/. Acesso em: 14/11/2016.