AEROMODELOS: SOLUÇÃO DIRIGIDA À MODELOS PARA DESIGN E IMPLANTAÇÃO AUTOMÁTICA DE SOFTWARE NA NUVEM

Franklin Magalhães Ribeiro Junior, Tarcísio da Rocha
Instituto Federal de Educação Ciência e Tecnologia do Maranhão (IFMA)
Universidade Federal de Sergipe (UFS)
franklin.mr3@gmail.com, tarcisiorocha@gmail.com

***Resumo*: A computação em nuvem reduz custos com hardware e software por tratar tudo como serviço. Contudo, para implantar um software na nuvem é preciso que o desenvolvedor tenha conhecimento técnico da arquitetura provedor de nuvem que deseja implantar o software. Nesta investigação foi elaborada uma abordagem para implantação automática de software na nuvem e também a concepção dos modelos UML como entrada da solução de implantação, de forma que possibilitou ao desenvolvedor a criação de modelos de implantação intuitivamente e com pouco esforço de aprendizado. Também foi definido um possível padrão para modelos de implantação de software em nuvem, onde os requisitos ficaram mais visíveis, visto que não existe a definição de um padrão universal de modelos nessa linha. Na pesquisa foi realizado um experimento com a solução, nele foi percebido que a solução apresentou um impacto positivo em relação à solução convencional no que se refere às métricas de manutenibilidade, apreensibilidade e redução da carga de trabalho do desenvolvedor na implantação de software em nuvem.**

***Palavras-chave:* UML design; Implantação de software; Computação em nuvem; Desenvolvimento de software baseado em modelos.**

***Abstract*: The cloud computing reduces hardware and software costs by treating all as a service. However, software deployment in cloud requires that developer to have technical knowledge of the cloud provider architecture which wants to deploy the software. In this research, we proposed an approach to automatic software deployment in cloud and the UML design models as input of the deployment solution, in a way that allowed the developer to create deployment models intuitively and with low costs to learning. Our solution turns possible a standard for cloud software deployment models, with that, requirements have become more visible since there is no universal standard to software deployment models in cloud. In the research, we did an experiment with the solution, in it was realized that the solution had a positive impact against the conventional solution, using metris: maintenance, apprehensibility and workload reduction to the software deployment in the cloud.**

***Keywords:* UML design; software deployment; cloud computing; model based software development.**

I. INTRODUÇÃO

Os recursos trazidos pela computação em nuvem, através da virtualização, reduzem os custos computacionais provenientes da necessidade de processamento de uma alta demanda de dados [13, 16]. Atualmente os recursos da nuvem se aplicam em soluções que compreendem diversas áreas do conhecimento.

Além da redução dos custos de infraestrutura, a computação em nuvem possui uma abordagem baseada na entrega de serviços[18]. Nela o cliente pode requisitar um serviço de software no provedor de nuvem, implantado pelo fornecedor do serviço ou implantar o seu próprio serviço de software na nuvem. Entretanto, implantar um sistema de software em um ambiente na nuvem requer a reconstrução de vários requisitos existentes, já que ambientes em nuvem possuem arquiteturas de software próprias [4].

Em sistemas onde o desenvolvimento é bastante custoso, o estabelecimento de uma abordagem de implantação automática de serviços e aplicações na nuvem, que reduz a carga de trabalho aos desenvolvedores é vantajoso.

Por isso, nesta pesquisa foi desenvolvida e experimentada uma solução de implantação automática de software na nuvem dirigida à modelos, nomeada de AeroModelos, cujo o uso de modelos na solução de implantação apresenta impactos positivos sobre as métricas [14, 17] de carga de trabalho, manutenibilidade e apreensibilidade.

O trabalho está organizado da seguinte maneira, na Seção 2 apresenta os requisitos de implantação de software na nuvem, a Seção 3 expõe os trabalhos relacionados, a Seção 4 analisa os mecanismos de implantação na nuvem, a Seção 5 explana a escolha da ferramenta CASE usada no trabalho, a Seção 6 apresenta a solução proposta, a Seção 7 detalha a arquitetura da solução, a Seção 8 mostra um estudo de caso com a solução, a Seção 9 apresenta um experimento com a solução a partir do estudo de caso e finalmente a Seção 10 apresenta as conclusões e trabalhos futuros.

II. IMPLANTAÇÃO DE SOFTWARE NA NUVEM

Para que o software execute corretamente numa arquitetura em nuvem, faz-se necessário a implantação deste e (ou) dos demais serviços que este software depende na nuvem, cujos serviços são compostos ou tratados como componentes [15].

Uma maneira mais eficiente da implantação de software em ambientes em nuvem é a automatização desse processo, já que detalhes dispensáveis de configuração poderiam ser abstraídos, além de reduzir a carga de esforço humano na implantação de software.

Ademais, a implantação automática de software na nuvem é vantajosa, devido à eliminação de tarefas manuais como: a configuração da pilha de software, a autenticação entre máquinas virtuais, a definição das dependências dos componentes de serviço e entre os serviços, a definição das dependências temporais e espaciais e finalmente, a análise dos recursos disponíveis nos ambientes em nuvem, a fim de

identificar se o nó na nuvem atende os requisitos mínimos de determinado serviço [11].

Dentre as características mais importantes na implantação automática em nuvem, destacam-se o uso de (i) Coordenadores: que gerenciam uma pilha de software [4], (ii) Pilha de Software: contém serviços que suportam uma aplicação, (iii) Máquina Virtual (VM): onde os serviços da pilha de software executam para suportar a aplicação [2] e (iv) nós clientes: que encapsulam a VM [10].

III. TRABALHOS RELACIONADOS

Nesta pesquisa, após realizada uma investigação bibliográfica acerca das abordagens de implantação automática de software em nuvem, foram encontrados e selecionados os trabalhos em [1], [3], [4], [6], [10], [11], [12] e [20], onde em cada proposta foi descrita e explanada nesta seção, no âmbito de analisar cada investigação relacionada.

A. *D&C Based Deployment*

O trabalho em [4] foi impulsionado pela necessidade de execução de uma aplicação que analisa a estrutura química de um fármaco, o QSAR, utilizada por um sistema multi-agente, chamado de Discovery Bus. Tendo em vista que, a arquitetura de fluxo do QSAR utiliza algoritmos de exploração exaustiva para decifrar estruturas químicas e que essa exploração exaustiva demanda grande carga de processamento, o software foi implantado em nuvem para redução dos custos de processamento.

O processo para a implantação funcionava da seguinte forma: foram definidos planos de implantação (nestes planos são definidas as dependências e configuração para cada aplicação) que mais tarde serão enfileirados em uma fila de armazenamento de serviços. Depois de criada a fila de serviços de software, cada serviço é enviado a um módulo Deployer Engine para que seja realizada a implantação de software na nuvem, como especificado previamente pelo plano de implantação, o qual define a instalação e plataforma de execução do software.

B. *SDO*

Em sua pesquisa, Li et al. [12] propuseram a implantação de software em um ambiente na nuvem e satisfizeram a implantação de serviços em uma gama de cenários, como o cenário de nuvens privadas, bursted clouds, federated cloud, multi-cloud e brokering cloud.

Os autores propuseram uma arquitetura para atender os requisitos da implantação de software, a qual foi nomeada de Service Deploy Optimizer (SDO), a qual implanta os componentes dos serviços a diferentes provedores de nuvem.

Os autores validaram a arquitetura proposta usando o OPTIMUS toolkit em três cenários de implantação em nuvem, o de nuvem privada, brokering cloud e bursting cloud. Por fim, concluíram que maior parte do tempo de implantação depende do cenário da nuvem.

C. *Wrangler*

Na pesquisa em [10], os autores descreveram e avaliaram um sistema chamado Wrangler, que objetiva a implantação de aplicações em um provedor na nuvem. Para realização da implantação de software na nuvem, o Wrangler possui como entrada um documento XML com as configurações da implantação na nuvem.

D. *Disnix*

Objetivando a homogeneidade do acesso a serviços apenas encontrados em dispositivos físicos de um hospital ligados a redes de topologias complexas e de diferentes políticas de segurança, como aparelhos de ressonância magnética, computadores, entre outros, em [3] os autores propuseram uma arquitetura para aplicações, em que os componentes de software são automaticamente implantados na nuvem, com uso de um modelo de declarativo chamado Nix e o Disnix.

Para garantir a confiabilidade da implantação de aplicações, os autores propuseram artifícios em sua proposta que garantiam que versões mais antigas de serviços (softwares) já implantados na nuvem não afetassem as novas versões do mesmo serviço (há um controle de versões na implantação de software).

E. *Vega*

Em suas pesquisas, os autores em [6] propuseram um framework nomeado de Vega, o qual foi implementado para a nuvem IBM Research Compute Cloud e objetiva a especificação dos requisitos da solução de implantação de software na nuvem usando XML.

Os autores relatam que o framework apresentado provê mecanismos que possibilitam aos administradores configurar as dependências entre as VMs e entre as aplicações (pilha de software). O Vega também gerencia recursos do hardware (configurações da máquina utilizada na nuvem) e de rede (hosts e políticas de segurança). Finalmente os autores afirmam que essa técnica possui uma falha, pois não funciona bem em cenários muito complexos.

F. *User-Level Deployment*

No âmbito de desacoplar o software da VM, os autores em [20] desenvolveram um framework para implantação de aplicações em nuvem, cuja ideia visa a redução do tempo e gastos com implantação. Esta proposta difere das demais abordagens apresentadas até o momento, pois nesta proposta o software não estaria pré-instalado em uma imagem de VM.

A solução consiste em prover um servidor central de distribuição para fornecer software sob demanda às VMs, ao invés de uma VM conter um software previamente instalado.

A VM (que conterá o software) é inicializada pelo servidor e nela é instalado o software que o cliente escolheu do repositório. O software escolhido seria transmitido por uma sequência de bits para uma VM local (que contém o SO instalado) com sobreposição de execução, que permitiria que não fossem alocadas várias VMs e, por sua vez, a não espera do carregamento de uma imagem de VM inteira para executar o software.

Finalmente os autores fizeram um protótipo da proposta, onde foram implantadas dez aplicações e perceberam que o sistema sofre perda de desempenho, pela carga de transferência de dados ao cliente.

G. Virtual Deployment Models (VDM)

Em [11] foi apresentada uma arquitetura que suporta a implantação de software dirigida a modelos, a técnica mostra quatro fases, tais quais a fase de criação de imagens virtuais (Virtual Appliance), o modelo de solução virtual (VSM), o modelo de solução virtual de implantação (VSDM) e o plano de implantação virtual da solução (VSDP). Konstantinou, et al. (2009) também explanaram as quatro fases da sua proposta, em (i) consta a atuação dos usuários denominados especialistas de domínio, os quais constroem aplicações virtuais implementadas em imagens virtuais pré-configuradas, (ii) o VSM que descreve os requisitos da implantação e a configuração da rede, (iii) a VSDM, que deve agregar as configurações de uma nuvem específica para que possa seguir a (iv) um plano de implantação (VSDP).

Devido ao plano de implantação (VSDP) operar em dois níveis, um de controle das VMs e outro de configuração da pilha de software, a complexidade nessa abordagem foi reduzida, já que foram abstraídos aspectos como problema de versão de software e compatibilidade do software com a nuvem. Finalmente os autores projetaram um protótipo da solução, com uso de XML para viabilizar a comunicação entre as imagens de VM e a abordagem dirigida a modelos, com a representação dos pacotes da aplicação virtual em open virtualization format (OVF).

H. MODAClouds

Na pesquisa em [1] foi apresentado o projeto MODAClouds que ainda está sendo desenvolvido com auxílio da *European Commission* como uma abordagem dirigida a modelos para o design e execução de aplicações em diferentes tecnologias de nuvens. A arquitetura do MODAClouds foi dividida em dois escopos o design da implantação do software e o monitoramento da execução do software na nuvem (quanto ao *QoS*).

A abordagem abarca o uso de uma IDE (própria do MODAClouds), onde através dela o usuário pode definir três níveis, cada um constituído por vários modelos [21] para implantação de software na nuvem: (i) o nível de modelos onde são detalhados as restrições, o modelo de requisitos do negócio da aplicação, o modelo de dados e o modelo de comportamento do serviço, nomeado de (CIM), (ii) o nível dos modelos cujos artefatos são representados de acordo com cada serviço definido em (CIM) e independentes da tecnologia da nuvem, nomeado de (CPIM) e (iii) o nível dos modelos específicos para o provedor de nuvem de acordo com cada componente em (CPIM), nomeado de (CPSM).

Os autores também relatam que o MODAClouds também é capaz de lidar com software legado, uma vez que é dividido por níveis de modelos (CIM, CPIM e CPSM), onde por exemplo para a tarefa de troca de um provedor de nuvem para outro seria apenas necessário a mudança no nível CPSM. Apesar do alto nível de abstração para a implantação do software na nuvem, parte do nível CPSM deve ser parcialmente implementado (em código) para que seja realizada a implantação na nuvem, além de exigir do desenvolvedor certo conhecimento da tecnologia da nuvem em que se deseja implantar o software.

IV. ANÁLISE DOS MECANISMOS DE IMPLANTAÇÃO

Com o objetivo de descobrir e analisar os mecanismos utilizados por soluções relacionadas à implantação automática de software na nuvem foram selecionadas algumas características relevantes das propostas apresentadas no estado da arte.

Os autores em [19] discutiram os mecanismos existentes para implantação de serviços, como mecanismos de implantação manual, por meio de scripts, linguagens de programação e baseado em modelos. Os autores perceberam que mecanismos de linguagem de programação e abordagem dirigida a modelos lidam melhor com os custos, porém, aumentam a complexidade da solução, como pode ser percebido na Figura 1.

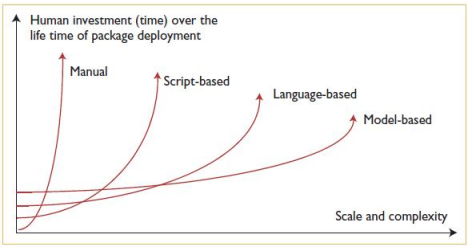

Figura 1. Mecanismos (Talwar, 2005)

Dentre os trabalhos relacionados, percebeu-se que o mecanismo Virtual Models [11] e MODAClouds [1] apresentam uma abordagem por implantação de software dirigido a modelos e com geração de código, como pode ser percebido na Tabela 1. Além disso, percebeu-se que D&C Based [4] apresentou alguns modelos para a implantação, porém apenas para restrições no processo de implantação e não na tarefa de implantação propriamente dita (além de não gerar código).

Observou-se que as soluções D&C Based [4], SDO [12] e UserLevel Deployment [20] utilizaram o mecanismo baseado em linguagem de programação, que em contraste com as abordagens Veja [6], Wrangler[10] e Disnix [3] apresentaram se mais vantajosas quanto ao investimento de tempo para o processo de implantação (ver Tabela 1).

As soluções das abordagens Veja [6], Wrangler [10] e Disnix [3] utilizaram mecanismos manuais ou script, o que exige mais tempo para o processo de implantação, ademais, de acordo com [9] e [19] estes mecanismos têm um menor grau de vantagens, porque aumentam os custos com relação ao código e aos esforços humanos.

Além disso, percebeu-se que as propostas Virtual Models [11] e Disnix [3] apresentaram mecanismos semiautomáticos para a implantação. Em outras palavras, nessas abordagens ainda há a necessidade de intervenção manual no processo de implantação.

Tabela 1. Características das Propostas contra a Solução AeroModelos.

	Mecanismos de Implantação				
Propostas	**Manual**	**Script**	**Linguagem**	**Modelos**	**Modelos Simples**
MODAClouds [1]				X	
Disnix [3]	X	X			
VDM [11]	X	X		X	
Vega [6]		X			
D&C [4]			X		
Wrangler [10]		X			
SDO [12]			X		
User-Level [20]			X		
AeroModelos (Solução Proposta)				**X**	**X**

Também foi descoberto que a proposta em MODAClouds[1] apresentou uma abordagem dirigida a modelos para a implantação de software, no entanto, esta abordagem requer certa compreensão do usuário final sobre detalhes da estrutura de computação em nuvem, além da grande carga de informação nos modelos demandada ao usuário para a implantação.

Nossa solução (AeroModelos) consiste em uma abordagem dirigida a modelos (a qual requer poucos modelos) para implantação automática de software na nuvem, cujo único conhecimento específico do desenvolvedor sobre a nuvem seja a chave de acesso e o nome do provedor, já que os detalhes mais específicos serão abstraídos. O objetivo dessa abordagem é a implantação de software em um nível de abstração mais elevado, no âmbito de reduzir o investimento humano e o tempo para a implantação, tanto quanto possível, já que o uso de uma abordagem com base em modelos é a melhor maneira de aumentar a produtividade do desenvolvedor [9].

V. ANÁLISE DAS FERRAMENTAS CASE

Uma das tarefas da proposta seria a interpretação dos arquivos gerados pelos diagramas UML é a conversão em código executável para a implantação na nuvem.

Para interpretar os diagramas de implantação do padrão UML, foram identificados e analisados os arquivos de saída de oito ferramentas CASE de modelagem UML de licença gratuita, dentre elas: ArgoUML [23], Dia [24], Gaphor[26], Modelio [25], NClass [28], StarUML [27], Umbrello[22] , VioletUML [29].

Foi descoberto que apenas três ferramentas, StarUML, Umbrello e VioletUML apresentaram arquivos de saída (diagramas UML) com metalinguagens de maior facilidade para a interpretação, por possuírem identificadores (IDs) não somente das coordenadas dos componentes visuais da UML, mas também IDs das associações entre os mesmos. Contudo, as ferramentas Umbrello e VioletUML não possuem diagramas de implantação como no StarUML (ver Quadro 1).

Ademais, foi percebido que as ferramentas Dia, NClass, Modelio, Gaphor e ArgoUML não apresentaram identificação clara entre as associações dos componentes.

Quadro 1. Características dos Arquivos de Saída das Ferramentas CASE

Ferramentas UML	**Diagrama de Implantação**	**Saída de Formato Textual**	**Identificação das Coordenadas**	**Componente com ID**	**Identificação das Associações**
ArgoUML [23]		X	X	X	
Dia [24]	X	X	X	X	
Gaphor [26]		X	X	X	
Modelio [25]	X				
NClass [28]		X	X	X	
StarUML [27]	**X**	**X**	**X**	**X**	**X**
Umbrello [22]		X	X	X	X
VioletUML [29]		X	X	X	X

Como pode ser percebido no Quadro 1, o *StarUML* atendeu a todas as necessidades da proposta. Por essa razão foi adotado o *StarUML* como ferramenta auxiliar na solução para o desenho dos modelos do AeroModelos.

VI. SOLUÇÃO DO AEROMODELOS

A implantação automática de serviços pode reduzir os custos com complexidade, além disso sabe-se que com o uso de modelos na implantação pode ser possível obter um ganho maior na produtividade e escalabilidade da implantação.

Esta seção explana como foram definidos os modelos de implantação, a arquitetura e a visão conceitual da proposta para a implantação automática de software na nuvem, além disso, também é apresentada a maneira como será realizada a interpretação dos modelos UML em adequação à proposta, já que para a solução são utilizados diagramas de deployment UML [15] como entrada.

A. Problemática

Em uma abordagem prática, suponha que o desenvolvedor deseje implantar uma aplicação na nuvem que dependa de um ou mais serviços, por exemplo. Para cumprir esta tarefa seria necessário que desenvolvedor: (i) adquirir uma chave de acesso ao provedor de nuvem que deseja implantar a aplicação, (ii) escolher uma máquina (nó) do provedor de nuvem para suportar a aplicação, (iii) realizar a configuração de uma máquina virtual, (iv) instalar da máquina virtual em um provedor de nuvem, (v) instalar de um sistema operacional

na máquina virtual e (iv) realizar a implantação dos serviços e aplicação na ordem de dependência dos serviços e da aplicação também desta máquina virtual.

B. *Funcionamento da Solução*

A solução AeroModelos propõe que inicialmente o desenvolvedor tenha implementado (caso necessário) os serviços que deseja implantar na nuvem, assim como ter criado os modelos UML de implantação (de software na nuvem), onde serão definidas todas as informações necessárias (máquinas virtuais, serviços, aplicações, dependências, sistemas operacionais das máquinas virtuais, repositório dos serviços e máquinas virtuais, banco de dados, provedor de nuvem, chaves de acesso) para a implantação apenas como forma de parâmetros (sem necessidade de codificar ou configurar nenhuma máquina virtual).

Após a criação e definição dos modelos com seus respectivos parâmetros de entrada, o usuário desenvolvedor utiliza como entrada no sistema os modelos e os serviços e então é disparada a implantação automática pelo sistema, nela ocorrem quatro etapas, sendo duas para a interpretação dos modelos UML, uma etapa da criação da pilha de software e outra para a geração automática de código. No que se refere as etapas de intepretação dos modelos de implantação UML, temos que (i) corresponde a interpretação de um modelo específico (que contém os elementos referentes ao provedor de nuvem, chave de acesso, instância da máquina virtual na nuvem e repositório) e (ii) a interpretação do modelo geral (que contém os elementos referentes às máquinas virtuais, sistema operacional, serviços, banco de dados, dependências e aplicações).

Para a criação da pilha de software temos que a pilha de software é definida pela pilha de dependências entre os serviços (que foram descobertos na etapa de interpretação do modelo geral).

Na primeira etapa o após disparada a implantação automática, o sistema inicialmente interpreta o modelo geral coletando seus dados através de um arquivo de formato similar ao XML, o "*.uml*", onde nele estão contidos os dados que o desenvolvedor especificou no modelo, instanciando lista de objetos máquina virtual, dependências, S.O. e serviços. Na etapa de criação da pilha de software, o sistema relaciona os serviços às respectivas máquinas virtuais e depois relaciona as dependências com os serviços de cada máquina virtual individualmente.

Ao finalizar essas duas etapas iniciais já foram coletados todos os dados referentes às máquinas virtuais e serviços, então, para coletar os dados restantes (como a qual provedor de nuvem a máquina virtual será instalada) é iniciada a etapa de interpretação do modelo específico, onde é também lido um arquivo de formato similar ao XML (formato *.uml*) para a coleta dos dados e finalmente após todos os dados coletados, é realizada a etapa de geração automática de código, onde o código é gerado e executado pelo sistema, para que se concretize a implantação automática de software na nuvem.

C. *Definição dos Modelos de Implantação*

No âmbito de buscar uma proposta implantação de software na nuvem, a solução tem como entradas dois diagramas de *deployment* UML: o primeiro é geral (conforme o exemplo da Figura 2) e independente de provedor de nuvem, apenas representando as máquinas virtuais (*VMs*), o sistema operacional e as dependências entre os serviços alocados a estas *VMs*, já o segundo é específico (conforme o exemplo da Figura 3) e possui aspectos específicos vinculados à infraestrutura da nuvem como, por exemplo, as chaves de acesso ao provedor.

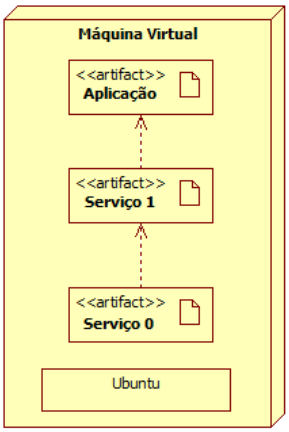

Figura 2. Modelo Geral

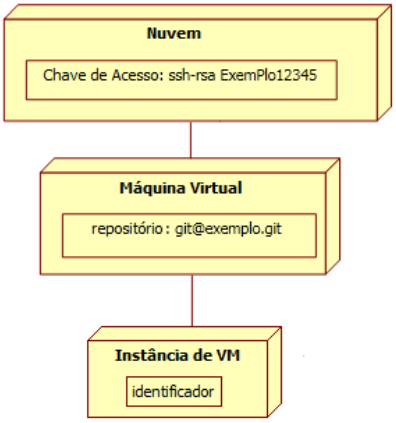

Figura 3. Modelo Específico

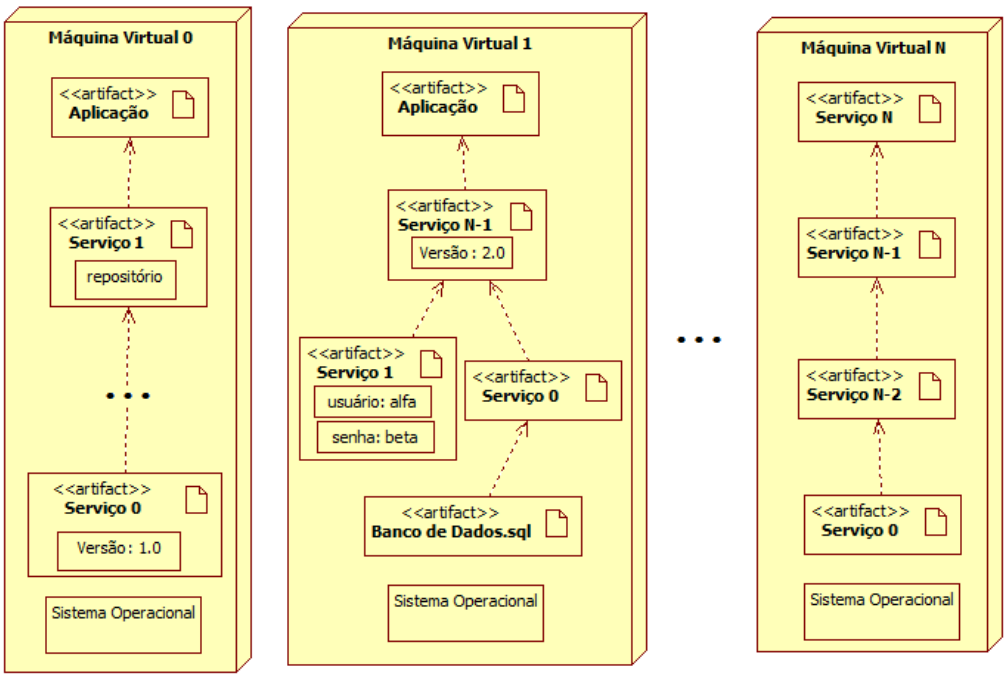

Figura 4. Exemplo do Modelo Geral com N Máquinas Virtuais

Pode-se perceber que o exemplo de modelo geral representado pela Figura 2 abrange todas as características necessárias (serviços, dependências entre serviços, sistema operacional, máquina virtual (VM), aplicação, banco de dados e campos opcionais para cada serviço, como por exemplo a versão do serviço, o diretório do serviço, entre outros) para a implantação do software independente do provedor de nuvem e que o modelo específico (Figura 3) coleta as informações necessárias para a implantação efetiva do software na nuvem, tais quais: chave de acesso, repositório dos serviços e aplicação da máquina virtual e instância de VM na nuvem.

Logo pode-se verificar que, para a efetiva modelagem do modelo específico, faz-se inicialmente necessária a criação do modelo geral, já que o modelo específico precisa das informações da máquina virtual, conforme a Figura 3.

A relação entre o modelo específico e o geral pode ser percebida de tal forma que no exemplo da Figura 3 o nó Máquina Virtual corresponde ao mesmo nó representado no exemplo da Figura 2, sendo o provedor de nuvem, o repositório e o banco de dados do modelo específico da Figura 3, correspondente ao nó Máquina Virtual da Figura 2.

Ao partir de uma solução prática (em grande escala), supondo que o desenvolvedor deseje a implantação de várias aplicações, o uso de modelos poderia tornar o trabalho de implantação de software menos árduo, pela redução de codificação e pelo aumento na produtividade da implantação, além de lidar com softwares legado, uma vez que o modelo geral é independente do provedor de nuvem, o que possibilita a migração da implantação para outro provedor de nuvem, como pode-se perceber na Figura 4.

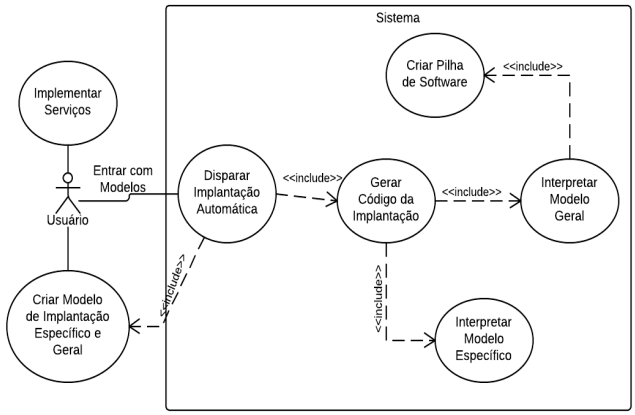

Figura 5. Diagrama de Casos de Uso do AeroModelos

D. Visão Geral da Solução

O diagrama de casos de uso da proposta pode ser observado na Figura 5 (como uma visão geral da solução) que visa ilustrar a minimização dos esforços humanos no processo de implantação. Nesse caso de uso, pode-se observar que a carga de trabalho é resumida na criação dos diagramas de *deployment* UML e na implementação dos serviços (caso necessário). Onde o usuário, apenas teria que utilizar como entrada do sistema os serviços implementados, os dois modelos de *deployment*, um geral (para *VMs* e serviços) e o outro específico (correspondente aos aspectos da nuvem escolhida).

O diagrama de casos de uso da Figura 5 é composto por sete casos de uso são eles:

- Implementar Serviços: no qual o desenvolvedor irá implementar os serviços e aplicação que deseja implantar no provedor de nuvem;
- Criar Modelo de Implantação Específico e Geral: que consiste na modelagem da implantação a ser realizada, onde estarão as informações necessárias para a implantação de cada aplicação;
- Disparar Implantação Automática: que possui uma visão geral do fluxo de atividades do sistema como um todo;
- Gerar Código da Implantação: que descreve as tarefas necessárias para a geração do código de configuração dos serviços e máquinas virtuais a serem implantadas;
- Interpretar Modelo Específico: que corresponde a interpretação e coleta de dados do modelo de implantação UML relacionado às características específicas do provedor de nuvem;
- Interpretar Modelo Geral: que corresponde a interpretação e coleta de dados do modelo de implantação UML relacionado aos serviços, aplicações e máquinas virtuais (e portanto que é independente de provedor de nuvem);
- Criar Pilha de Software: que é responsável pela criação da pilha de software, a qual representa a sequência em que os serviços devem ser instalados na máquina virtual, já que existe uma ordem de dependência entre os serviços, ou seja, um serviço A que dependa de um serviço B, somente executará corretamente quando B for instalado e executar antes de A.

Além disso, dentre os casos de uso do diagrama, dois deles ("Implementar Serviços " e também "Modelar Modelos de Implantação Específico e Geral") pertencentes somente ao escopo do desenvolvedor e cinco pertencem ao escopo do sistema, sendo em que apenas um caso de uso nomeado de "Disparar Implantação Automática de Software", o desenvolvedor poderá somente inserir os modelos criados no caso de uso "Criar Modelos de Implantação Específico e Geral".

VII. ARQUITETURA DO AEROMODELOS

A implantação arquitetura proposta está dividida em três visões (de sistema, local e remota) e é composta por cinco módulos, nomeados de Controlador, *Associador*, Empilhador, Alocador e Preparador.

Na Figura 6 pode ser vista a arquitetura do sistema proposto para implantação dirigida a modelos de software em nuvem. Dentre as visões estão:

- A visão de sistema: que mostra os cinco módulos do sistema (Controlador, *Associador*, Empilhador, Alocador e Preparador);
- A visão local: que corresponde à visão do usuário, ou seja, à interação humano-computador (que usa como entrada passwords e os diagramas de *deployment* UML);
- A visão remota: que inclui a criação de uma instância do servidor Chef [5] na nuvem e o estágio final da implantação, correspondente à instalação do software na nuvem.

A. *Módulo Controlador*

Na arquitetura, o módulo de Controlador gerencia os outros quatro módulos, atuando como um intermediário na comunicação entre os módulos e é responsável por:

- Enviar os serviços obtidos do *Associador* ao Empilhador e os relacionamentos de dependência entre os serviços (pilha de software), do Empilhador para o Preparador;
- Habilitar a comunicação do Alocador com a nuvem;
- Informar ao Alocador sobre a criação da instância do servidor na nuvem;
- Informar ao Preparador que sistema operacional deve ser instalado na VM;
- Informar ao Preparador que pilha de software deve ser alocado na respectiva VM;
- Gerenciar a ordem da transmissão de informação e realização de tarefas entre os módulos.

B. *Módulo Associador*

Para a interpretação do código em UML, foi implementado na solução o módulo Associador, que utiliza os modelos Geral e Específico (modelados pelo desenvolvedor) como entrada na solução proposta.

A rotina de execução do módulo Associador foi dividida em duas etapas: (i) Interpretação do Modelo Geral e (ii) Interpretação do Modelo Específico, onde em cada etapa foram usados arquivos de entrada (metadados dos modelos UML), os quais lidos linha a linha através de um "*BufferedReader*".

A cada linha lida no arquivo o algoritmo da solução busca pela *substring* "*OwnedViews*", que pode referenciar um componente UML do tipo artefato, nó, associação ou dependência. Quando a linha do arquivo que contém a *substring* "*OwnedViews*" é encontrada, o método busca por uma outra substring na mesma linha do arquivo, onde essa *substring* pode ser: (i) "*UMLNodeView*" que representa um Nó, (ii) "*UMLArtifactView*" que representa um artefato, (iii) "*UMLAssociationView*" que representa uma associação (apenas usado na interpretação do Modelo Específico) e (iv) "UMLDependencyView" que representa uma dependência. O exemplo desses metadados pode ser percebido na Figura 7.

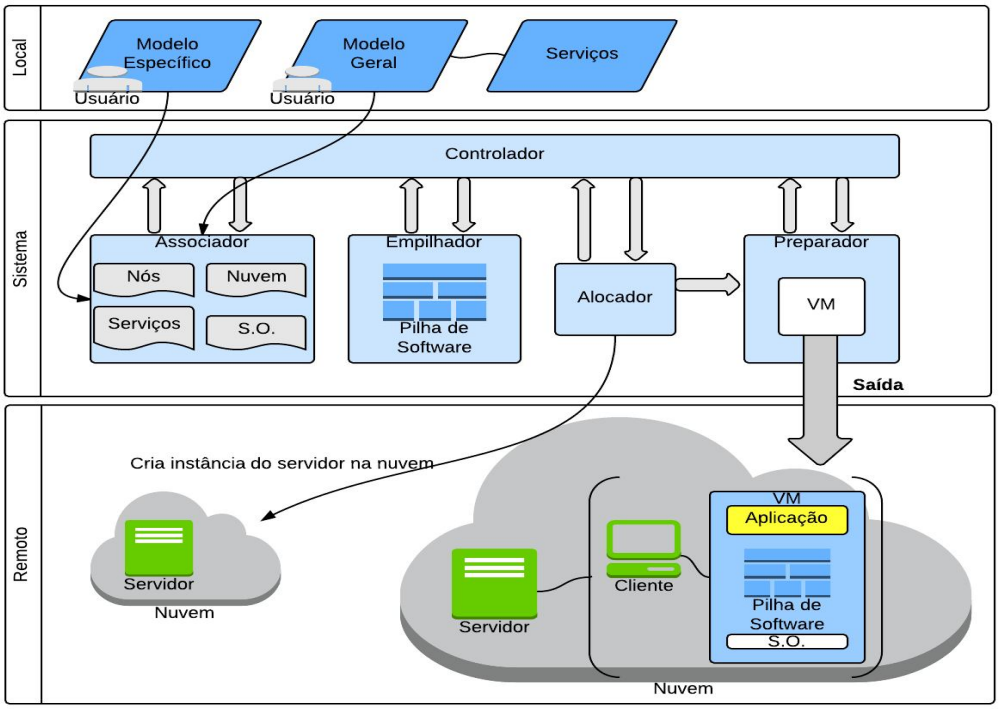

Figura 6. Arquitetura da Proposta

Depois de coletados os atributos referentes aos componentes UML são instanciados objetos (com seus respectivos atributos) e criadas listas desses objetos para cada tipo de componente UML, como listas de objetos nós, máquinas virtuais, provedores de nuvem, instância de máquina, serviços (artefatos), sistemas operacionais e atributos dos serviços.

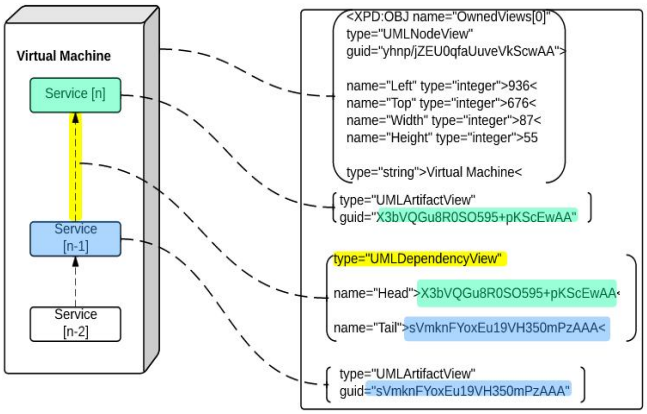

Figura 7. Exemplo de Interpretação do modelo UML pelo Módulo Associador.

C. *Módulo Empilhador*

Para definir a pilha de software (dependência entre os serviços de software) e também definir os serviços pertencentes às suas respectivas máquinas virtuais foi codificado, a partir da solução proposta por esta pesquisa, o módulo Empilhador.

1) *Relacionamento de Dependências com Serviços*

Para relacionar as dependências com os serviços foram utilizadas duas listas de objetos, obtidas a partir do módulo Associador. Na Figura 8 pode se notar que cada objeto da lista de objetos "Dependência" possui os atributos "head" e "tail" correspondentes aos identificadores dos serviços Service[b] e Service[a] respectivamente, onde Service [b] depende de Service[a]. Além disso, os objetos Artefato (que são os Serviços) possuem o seu próprio identificador (guid).

O algoritmo deste módulo então varre toda a lista de objetos dependência, "getListaDependencia()", para cada artefato. Quando o identificador (guid) do objeto artefato em questão for igual ao identificador do atributo "head" de um objeto na lista de dependências, então o vetor de atributos "dependeDe" do objeto artefato (serviço) recebe o atributo "tail" do objeto dependência em questão. Com isso um serviço obterá o serviço que ele depende para ser executado. Este mesmo processo de forma inversa é realizado para a obtenção do identificador, por exemplo, de um serviço pai que dependa

de um serviço filho, onde o filho armazenará o ID do serviço pai em uma variável denominada "anterior" (usada mais tarde para a criação da pilha de software).

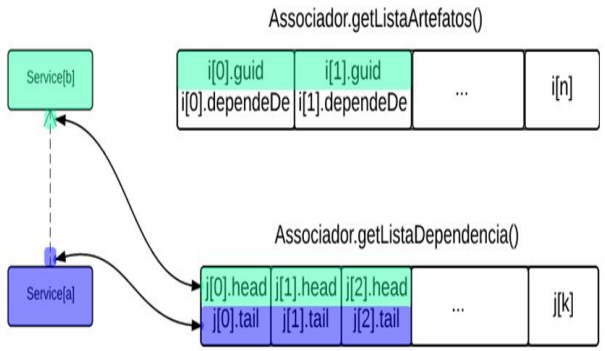

Figura 8. Relacionamento de Dependências com seus respectivos Serviços.

2) Relacionamento de VMs com Serviços

Para a estabelecer a relação entre os serviços que estão dentro da envoltória de suas respectivas máquinas virtuais, foi implementado um método que varre uma lista de máquinas virtuais (nós) para cada serviço (artefato), onde para cada serviço é verificado se o serviço pertence a envoltória da VM (ou seja, se no modelo as coordenadas X e Y do serviço estão contidas na envoltória do objeto máquina virtual).

3) Criação da Pilha de Software

Depois do processo de descoberta dos serviços com suas respectivas máquinas virtuais, cada máquina virtual possuirá uma lista de serviços com suas dependências, porém desordenados e sem ligação direta uns com os outros, por isso faz se necessária a criação de uma pilha de software para cada máquina virtual. Esta pilha de software é composta pelos serviços e aplicações na ordem em que devem ser instalados, para que por exemplo um serviço *A* que depende de um serviço B, seja implantado apenas após a implantação do serviço B.

Para a criação da pilha de software de cada máquina virtual individualmente foi implementado um algoritmo baseado no algoritmo de ordenação topológica [7]. A ideia da solução consiste em montar um grafo, cujo os nós desta estrutura de dados são os objetos *serviços* (contidos na lista de serviços de cada VM), onde para cada serviço há uma variável chamada de "*anterior*" que contém o valor do identificador (ID) do serviço pai (o serviço a que é dado o suporte), o identificador (ID) do próprio serviço em questão e uma lista de dependências nomeada de "*ListaDependeDe*[]" (lista de serviços que o serviço em questão depende).

Logicamente, o grafo de serviços que serve de base para a criação da pilha de software, já foi gerado quando foi executado o algoritmo de relacionamento entre dependências e serviços (ver subseção 1), através dele, cada serviço recebeu um valor para o atributo "*anterior*" e a lista dependências de serviços "*ListaDependeDe*[]". Logo, o grafo apenas precisa de um ponto de partida que inicia com uma busca pela lista de serviços, no âmbito de encontrar o serviço raiz (a aplicação) que possui um valor nulo na variável "*anterior*". Um exemplo do grafo da solução pode ser percebido na Figura 9.

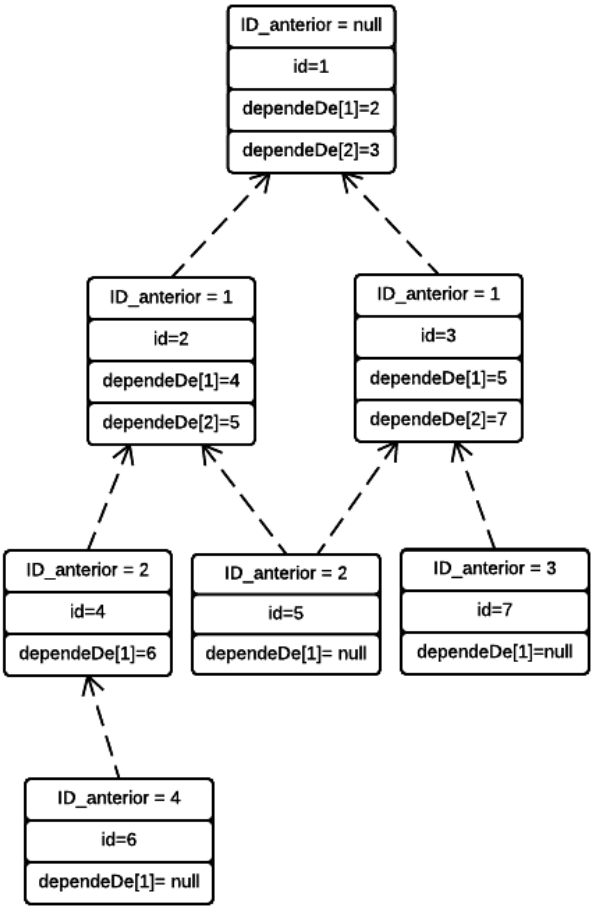

Figura 9. Exemplo da Estrutura Lógica do Grafo de Serviços

Depois que identificado o serviço raiz do grafo, foi implementado um algoritmo que realiza uma busca em profundidade pelo serviço que possui a lista de dependências ("*dependeDe*[]") com valor nulo (vazia). Quando encontrado o serviço que possui a lista "*dependeDe*[]" vazia, ele é empilhado na pilha de software e o serviço pai (o seu anterior) é buscado.

Em seguida, da lista de dependências do serviço pai é removida a dependência que ele tinha com o serviço filho, que acabou de ser empilhado. Então é verificado se este serviço pai possui mais algum serviço na sua lista de dependências. Em caso afirmativo o mesmo processo é realizado (até que a

lista do serviço pai fique vazia e todos os filhos sejam empilhados) e, caso a lista de dependências do serviço pai seja nula (vazia), então ele é empilhado e o seu serviço pai é buscado e assim sucessivamente, até que o algoritmo retorne à raiz e a raiz não possua nenhuma dependência.

D. Módulo Alocador

Após a interpretação do Modelo Específico, faz se necessária a alocação de (s) provedor (es) de nuvem às máquinas virtuais e o estabelecimento das instâncias de máquinas (machine image) na nuvem com as máquinas virtuais que se deseja implantar.

O módulo Alocador utiliza como parâmetro de entrada (tudo que foi obtido a partir do módulo Associador que são) todas as listas de objetos coletados do Modelo Específico, são elas: lista de nós, cujos nós podem ser máquinas virtuais, provedores de nuvem ou imagens de máquinas virtuais e os atributos do tipo "container" (PartUML) que são atributos dos objetos nós e podem representar o repositório dos serviços nas máquinas virtuais, o identificador da imagem de VM e a chave de acesso ao provedor de nuvem.

Tendo em vista que o módulo Alocador recebe uma lista de nós contendo vários tipos de nós em diferentes categorias e que os atributos dos nós como "PartUML" também possuem diferentes categorias, foi implementado para este módulo uma classe chamada "Alocador" que realiza quatro etapas: (i) a correlação dos atributos da lista "containers" com os objetos da lista de nós, (ii) a separação das listas de nós em categorias, tais quais: nuvem, imagem de VM e VM, (iii) o relacionamento do(s) objeto(s) provedor(es) de nuvem com a lista de objetos de máquinas virtuais e (iv) o relacionamento da lista de imagem(ns) de VM(s) com a lista de objetos de máquinas virtuais.

E. Módulo Preparador

Após coletados todos os dados e atributos pelo módulo *Associador* acerca dos serviços, máquinas virtuais, provedores de nuvem, banco de dados, imagem de VM, repositório da VM e após processados os dados pelos módulos Alocador e Empilhador para obtenção dos relacionamentos entre máquinas virtuais e serviços, provedores de nuvem, instância de VM e pilha de software, fica possibilitada a etapa de geração do código de implantação de software na nuvem.

No Preparador são gerados dois arquivos de saída para cada máquina virtual modelada pelo desenvolvedor no Modelo Geral e Específico, onde o primeiro arquivo corresponde a um script de implantação contendo o código executável para implantação e o segundo as regras da aplicação e serviços. Os códigos gerados para a implantação foram escritos pelo Preparador em linguagem ruby e seguem os padrões da plataforma Chef [20].

1) Geração do Código Executável da Implantação

A geração do arquivo contendo o código executável foi implementado em cinco partes: (i) a criação ou sobrescrita do arquivo de implantação, (ii) o cabeçalho do arquivo, contendo o caminho das definições ("cookbooks") dos serviços, (iii) os nomes e definições dos serviços e da aplicação, (iv) uma chamada _as regras dos serviços e aplicação (que estão definidas no outro arquivo que será gerado mais tarde pelo Preparador) e (v) o código executável da implantação, que contêm a especificação da nuvem, imagem de máquina virtual, sistema operacional, pilha de software e aplicação.

2) Geração do Código das Regras da Aplicação e dos Serviços

A geração do código do arquivo correspondente ás regras da aplicação foi implementada em cinco partes: (i) a criação ou sobrescrita do arquivo referente regras dos serviços, (ii) o cabeçalho do arquivo, composto pelo nome da aplicação e uma indicação de que ele representa as regras, (iii) o corpo do arquivo com o nome da aplicação, o repositório da aplicação e dos serviços, chave de acesso ao provedor de nuvem e a URL do banco de dados da aplicação, (iv) opcionalmente as regras dos serviços, contendo por exemplo a versão de cada serviço, porta, host, senha e usuário e (v) a nomenclatura de cada serviço e aplicação que serão executados na máquina virtual, na ordem em que estão na pilha de software.

VIII. Estudo de Caso da Solução

Para a visualização do código de saída gerado pela solução desta pesquisa foi implantado um software (usado pela indústria) na nuvem com o uso da solução proposta por esta pesquisa, cujo software implantado foi um e-commerce, produto da empresa SWX Softwares. Para a implantação deste software foram criados os modelos Geral e Específico contendo todos os requisitos necessários para a implantação.

No Modelo Geral foram descritos os requisitos máquina virtual com o nome de 14Commerce, a aplicação 14commerce-loja-virtual, os serviços RDS (servidor de banco de dados), WordPress (sistema de gerenciamento de conteúdo para Web), Nginx que é um servidor Web (correlato ao apache + MySql) e a URL do banco de dados, além disso foram estabelecidas as dependências entre os serviços e modelada a descrição do sistema operacional (Ubuntu) e especificadas as versões, repositório e host para cada serviço (quando necessário). O modelo geral que foi criado para a implantação pode ser percebido na Figura 10.

No modelo Específico foram descritos o provedor de nuvem que foi o *Elastic Cloud 2*, o repositório ("*41_14commerce-1.5.0-rc02-02.zip*") em que se encontram os serviços e aplicação da máquina virtual (14commerce) e a imagem de VM (*t1.micro*) com o identificador ("ami-967edcff") da imagem na nuvem, além das associações entre a instância da VM, provedor de nuvem e máquina virtual. O modelo Específico criado pode ser percebido na Figura 11.

Após o uso dos modelos Geral e Específico como entrada na solução foram gerados os arquivos que contém as regras da aplicação e serviços e o código executável da implantação que podem ser percebidos respectivamente nos códigos das Figuras 12 e 13.

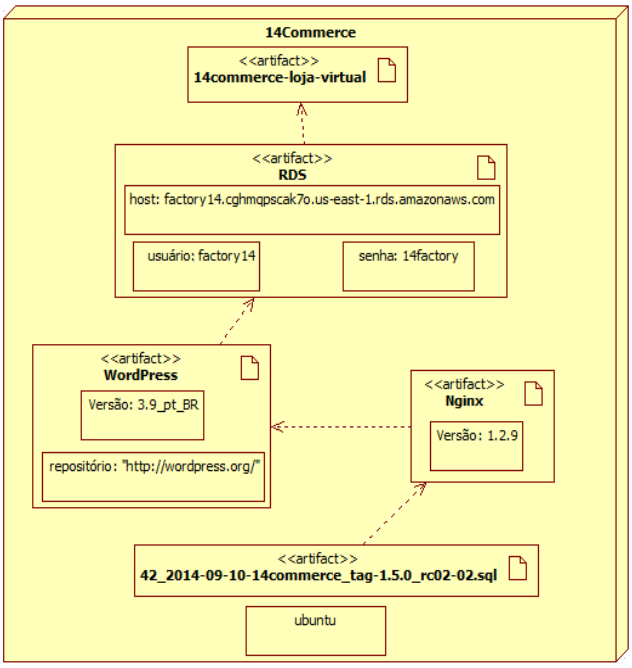

Figura 10. Modelo Geral para a Implantação do e-commerce *"14-commerce-loja-virtual"*

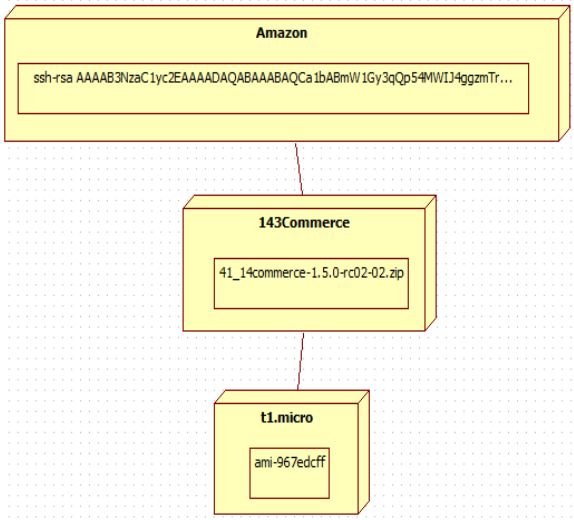

Figura 11. Modelo Específico do e-commerce


```
{
  "name": "14S_14commerce-loja-vitual",
  "description": "",
  "json_class": "Chef::Role",
  "default_attributes": {

  },
  "override_attributes": {
    "app": {
      "name": "14S_14commerce-loja-vitual",
      "repository_url": "",
      "access_key": "ssh-rsa
          AAAAB3NzaC1yc2EAAAADAQABAAABAQCa1bABmW1Gy3q
4ggzmT+dGVzJLzXOEASxdCeyFucRuX+
    jxTpfrpFM7wjbpB3hlapQu4r6SxNeBymHSLUPfYCN7onj5Ll9
1iJgdgmcxGY3BIgBmAWJyVdK3UUOiIBWKL22UMtYgcLCaPU0GlKX
    miSuFY1TtOeUt",
  "tag": "",
  "database_url":
    "42_2014-09-10-14commerce_tag-1.5.0_rc02-02.sql",
  "source_url": "41_14commerce-1.5.0-rc02-02.zip",
  "database": {
   "host":
   "factory14.cghmqpscak7o.us-east-1.rds.amazonaws.com",
    "user": "factory14",
    "password": "factory14"
  },
      "nginx": {
            "version": "1.2.9"
  },
      "wordpress": {
        "version": "3.9_pt_BR",
            "repourl": "http://wordpress.org/"
      }
    }
  },
  "chef_type": "role",
  "run_list": [
    "recipe[14Base]",
    "recipe[NginPressRDS]",
    "recipe[14S_14commerce-loja-vitual]"
  ],
  "env_run_lists": {

  }
}
```


Figura 12. Código das Regras de Implantação gerado pela Solução

```
berksfile:
  path: 'cookbooks-plataform/NginPressRDS/Berksfile'
cookbooks:
- NginPressRDS:
- 14commerce-loja-virtual:
- 14Base:
roles:
- NginPressRDS:
- 14commerce-loja-virtual:
- 14Base:
knife:
  - ec2 server create:
    - -I ami-967edcff -x ubuntu -f t1.micro -r
      'role[NginPressRDS], role
      [14commerce-loja-virtual], role[14Base]'
```

Figura 13. Código Executável da Implantação gerado pela Solução

IX. EXEPERIMENTAÇÃO

Neste contexto, esta pesquisa focou a análise da abordagem dirigida a modelos para a implantação automática de software na nuvem, com finalidade de avaliar o uso dos modelos da solução, com respeito à usabilidade, aprendizado e esforço de implantação do software na nuvem, do ponto de vista de desenvolvedores de serviços de software, no contexto da indústria.

É válido ressaltar que no Estado de Sergipe, onde foi realizado o experimento, apenas foi encontrada uma empresa que lida com a implantação de software na nuvem, por isso a experimentação foi realizada apenas como a aplicação de um estudo de caso na empresa SWX Softwares no parque tecnológico do Estado de Sergipe.

A. *Métricas*

Para o experimento, foram definidas métricas (quantificáveis) de avaliação, já que uma avaliação de software nesse escopo é subjetiva. Baseado nas métricas para avaliação de software estabelecidas em [14] e [17] foram encontradas as métricas:

- Apreensibilidade (facilidade de aprendizagem)
- Carga de Trabalho
- Consistência
- Atratividade
- Manutenibilidade

B. *Preparação do Experimento*

Para o experimento participaram 4 desenvolvedores empregados em uma empresa de software que realiza a implantação dos seus produtos na nuvem, todos os participantes com experiência na área. No estudo, foi solicitado aos participantes que utilizassem o software da Solução desenvolvida por esta pesquisa, para que fosse implantado um serviço de software no provedor de nuvem da *Amazon Web Services* (AWS).

Os participantes foram treinados acerca de como deveriam realizar a implantação com o uso da solução dirigida por modelos. Com isso, eles ficaram cientes de que para a implantação de software na nuvem utilizariam a ferramenta auxiliar StarUML para modelar os modelos Geral e Específico (como entrada para a implantação de software).

Foi utilizado um questionário aos participantes do experimento no âmbito de se coletar dados referentes às métricas de Apreensibilidade, Carga de Trabalho, Consistência, Atratividade e Manutenibilidade de implantação de software na nuvem.

O questionário foi utilizado em quatro escopos, o primeiro escopo referente ao nível de experiência do desenvolvedor de serviços, o segundo e terceiro relacionado apenas à Solução proposta, ou seja, sem uso de comparações com outras soluções e o quarto com intuito de comparar a solução proposta com a solução atualmente utilizada na empresa SWX Softwares através do uso das métricas.

Escopo 1: Relacionado apenas a Solução com avaliação das métricas – (respostas em escala de 1 a 5)

P1. *Você usaria a solução rotineiramente? (Atratividade)*

P2. *Com o uso da solução ficam mais claros os conceitos de como funciona a implantação de software na nuvem? (Consistência)*

Escopo 2: Comparativo e objetivo com uso das métricas – (respostas em escala de 1 a 5)

P1. *A proposta trazida pela solução de implantação dirigida a modelos é de fácil assimilação? (Apreensibilidade)*

P2. *A proposta trazida pela solução convencional é de fácil assimilação? (Apreensibilidade)*

P3. *Em pequena escala, 1 a 3 máquinas virtuais, a solução com o uso de modelos reduziu o esforço de implantação de software na nuvem? (Carga de Trabalho)*

P4. *Em pequena escala, 1 a 3 máquinas virtuais, a solução sem o uso de modelos reduziu o esforço de implantação de software na nuvem? (Carga de Trabalho)*

P5. *Em grande escala, mais de 4 máquinas virtuais, a solução com o uso de modelos reduziu o esforço de implantação de software na nuvem? (Carga de Trabalho)*

P6. *Em grande escala, mais de 4 máquinas virtuais, a solução sem o uso de modelos reduziu o esforço de implantação de software na nuvem? (Carga de Trabalho)*

***P7**. A proposta trazida pela solução de implantação dirigida a modelos é bem documentada e de fácil manutenção? (Manutenibilidade)*

***P8**. A proposta trazida pela solução de implantação por meio de codificação é bem documentada e de fácil manutenção? (Manutenibilidade)*

C. Execução do Experimento

Na execução do experimento os participantes elaboraram os modelos Geral e Específico para implantar um serviço de software na AWS, cujos modelos foram utilizados como entrada na Solução proposta por esta pesquisa.

O software que foi implantado para a execução do experimento foi o 14-Commerce, um serviço web de loja virtual da empresa SWX. Esta aplicação web possui dependências como, os serviços Nginx, WordPress e RDS (*Relational Database Service*) com seus respectivos modelos Geral e Específico, como podem ser observados nas Figuras 10 e 11.

D. Resultados e Análise

Após a execução da solução pelos desenvolvedores foi obtido o serviço e-commerce implantado na nuvem AWS e aplicado um questionário aos desenvolvedores participantes afim de se obter as métricas de Apreensibilidade, Carga de Trabalho, Consistência, Atratividade e Manutenibilidade.

O questionário possuiu dois escopos, sendo o primeiro para medir a atratividade e a consistência da solução com o código na visão do desenvolvedor e o segundo com intuito de realizar um comparativo das demais métricas obtidas pela solução com uso de modelos contra a solução manual.

Todos os participantes tinham conhecimento prévio da UML e possuíam experiência de com implantação de software na nuvem.

Para os escopos 1 e 2 do questionário cada pergunta correspondeu a uma das métricas que se desejou analisar, onde para cada alternativa de uma pergunta foi atribuído um peso relacionado à métrica que a pergunta correspondia, cujo peso tinha valor de 1(um) a 5(cinco), onde quanto mais próximo do valor 5(cinco) mais fortemente correlacionado à métrica a alternativa estaria.

Após coletados os dados referentes ao Escopo 1 do questionário foram obtidos os resultados das métricas de Atratividade e Consistência dispostos na Tabela 2, onde para cada métrica foi obtido um valor numérico correspondente a uma média de acordo com os pesos atribuídos por cada participante.

Tabela 2. Dados do Escopo 1

Métricas	Média Ponderada dos Pesos
Atratividade	3
Consistência	4.5

Com as médias obtidas pelo Escopo 1 pode-se perceber que para o usuário desenvolvedor a Solução apresentou Consistência enquanto solução de implantação de software na nuvem em um nível de 90% e Atratividade de uso para os usuários em 60%.

No caso da moderada aceitação no que se refere a Atratividade deve-se ao fato de que a interface deveria ser mais customizada, já o fato de a consistência ter tido de 90% de aceitação deveu-se ao fato de que para os usuários a solução por modelos em alguns casos mais específicos ainda possui fatores limitantes, embora os participantes tenham respondido que a solução por modelos seja fiel aos requisitos de implantação de software na nuvem.

Depois de aplicado o questionário para cada participante foram obtidos os dados referentes ao Escopo 2. Foi realizada a coleta e análise dos dados, onde para cada métrica foram obtidos os valores que podem ser percebidos na Tabela 3. Na Tabela 3 é possível perceber os resultados com o uso da solução e sem o uso da solução.

Tabela 3. Resultado com e sem uso da Solução Proposta

Métricas	Média Ponderada dos Pesos	
	Com uso da Solução Proposta	Sem a Solução Proposta
Apreensibilidade	4.25	2.75
Redução da Carga de Trabalho (Pequena Escala)	3.75	2.25
Redução da Carga de Trabalho (Grande Escala)	4.25	3
Manutenibilidade	4	2.5

Analisada a Tabela 3 pode se perceber que segundo os participantes, com o uso de modelos para a implantação de software na nuvem, todas as médias das métricas de apreensibilidade, redução nas cargas de trabalho em grande e pequena escala de implantação e a capacidade de manutenibilidade no que se refere à implantação de software na nuvem se apresentaram maiores que as médias dessas mesmas métricas com relação ao uso da solução convencional por meio de codificação. Fato que torna se mais claro como pode-se perceber no gráfico da Figura 14.

No gráfico da Figura 15 pode-se observar em pontos percentuais o impacto na diferença dos pesos atribuídos pelas médias de cada participante com relação ao uso e não uso da solução por modelos para a implantação de software na nuvem. Pôde-se observar que em média nos participantes com relação a cada uma das métricas de manutenibilidade, apreensibilidade e redução da carga de trabalho em pequena escala de implantação foi percebido um impacto positivo de 30% em pontos percentuais com o uso da solução e que em relação à redução na carga de trabalho em grande escala de implantação foi percebido um impacto positivo de 25%.

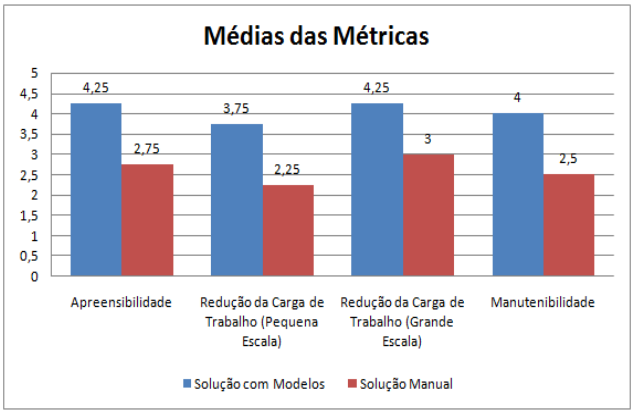

Figura 14. Gráfico de Médias das Métricas com e sem uso da Solução.

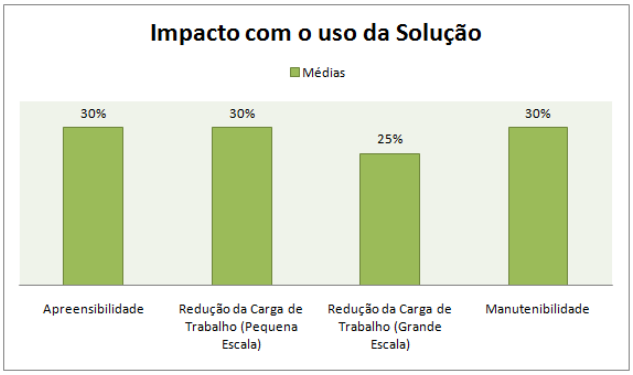

Figura 15. Gráfico do Impacto nas Métricas com uso da Solução

X. Conclusão

Nesta pesquisa foi apresentada uma abordagem dirigida a modelos para implantação automática de software na nuvem, (que utiliza diagramas UML como entrada) no âmbito de prover uma redução na carga de trabalho para os desenvolvedores. Também foi definido um possível padrão para modelos de implantação de software em nuvem (consistente com o código), onde os requisitos ficaram mais visíveis, visto que não existe a definição de um padrão universal de modelos nesse escopo.

Na investigação foi realizado um experimento envolvendo uma situação real em parceria com a empresa SWX Softwares. Como resultado do experimento da solução desenvolvida por esta investigação, percebeu-se que houve um ganho com a redução na carga de trabalho em média de pelo menos 25% para os desenvolvedores, além do impacto positivo de 30% de aumento nas métricas de apreensibilidade e manutenibilidade da solução de implantação de software na nuvem por modelos contra a solução convencional.

Como trabalho futuro espera-se desenvolver uma interface gráfica customizada para a criação dos modelos UML, visto que a ferramenta StarUML, utilizada como apoio para a criação dos modelos possui componentes gráficos pouco específicos de acordo com o que foi projetado para o ideal de implantação de software na nuvem, podendo em alguns casos comprometer a assimilação dos modelos para o usuário desenvolvedor. Ademais será investigada uma maneira de possibilitar que a solução gere a implantação automática de uma mesma máquina virtual em dois ou mais provedores de nuvem, inclusive em nuvens híbridas.

Referências

[1] Ardagna, D., Di Nitto, E., Casale, G., Petcu, D., Mohagheghi, P., Mosser, S., Matthews, P., Gericke, A., Ballagny, C., D'Andria, F., et al.: Modaclouds: A model driven approach for the design and execution of applications on multiple clouds. In: Proceedings of the 4th International Workshop on Modeling in Software Engineering. pp. 50–56. IEEE Press (2012)

[2] Armstrong, D., Djemame, K., Nair, S., Tordsson, J., Ziegler, W.: Towards a contextualization solution for cloud platform services. In: Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on. pp. 328– 331. IEEE (2011)

[3] Van der Burg, S., De Jonge, M., Dolstra, E., Visser, E.: Software deployment in a dynamic cloud: From device to service orientation in a hospital environment. In: Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing. pp. 61–66. IEEE Computer Society (2009)

[4] Cala, J., Watson, P.: Automatic software deployment in the azure cloud. In: Distributed Applications and Interoperable Systems. pp. 155–168. Springer (2010)

[5] Chef Software, I.: Chef - code. https://www.chef.io/, (Visitado em 01/02/2017)

[6] Chieu, T., Karve, A., Mohindra, A., Segal, A.: Simplifying solution deployment on a cloud through composite appliances. In: Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on. pp. 1–5. IEEE (2010)

[7] Cormen, T.H.: Introduction to algorithms. MIT press (2009)

[8] Dudin, E., Smetanin, Y.G.: A review of cloud computing. Scientific and Technical Information Processing 38(4), 280–284 (2011)

[9] Fazziki, A.E., Lakhrissi, H., Yetognon, K., Sadgal, M.: A service oriented information system: a model driven approach. In: Signal Image Technology and Internet Based Systems (SITIS), 2012 Eighth International Conference on. pp. 466–473. IEEE (2012)

[10] Juve, G., Deelman, E.: Automating application deployment in infrastructure clouds. In: Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on. pp. 658–665. IEEE (2011)

[11] Konstantinou, A.V., Eilam, T., Kalantar, M., Totok, A.A., Arnold, W., Snible, E.: An architecture for virtual solution composition and deployment in infrastructure clouds. In: Proceedings of the 3rd international workshop on Virtualization technologies in distributed computing. pp. 9–18. ACM (2009)

[12] Li, W., Svard, P., Tordsson, J., Elmroth, E.: A general approach to service deployment in cloud environments. In: Cloud and Green Computing (CGC), 2012 Second International Conference on. pp. 17–24. IEEE (2012)

[13] Muthunagai, S., Karthic, C., Sujatha, S.: Efficient access of cloud resources through virtualization techniques. In: Recent Trends In Information Technology (ICRTIT), 2012 International Conference on. pp. 174–178. IEEE (2012)

[14] Nielsen, J.: Usability engineering. Elsevier (1994)

[15] OMG: Uml 2.4.1. http://www.omg.org/spec/UML/2.4.1/, (Acesso em 01/06/2016)

[16] Salapura, V.: Cloud computing: Virtualization and resiliency for data center computing. In: Computer Design (ICCD), 2012 IEEE 30th International Conference on. pp. 1–2. IEEE (2012)

[17] Santos, R.C.: Revis˜ao das m´etricas para avalia¸c˜ao de usabilidade de sistemas (review of the metrics for evaluating the usability of systems). In: Congresso Internacional GBATA (2008)

[18] Savu, L.: Cloud computing: Deployment models, delivery models, risks and research challenges. In: 2011 International Conference on Computer and Management (CAMAN) (2011)

[19] Talwar, V., Milojicic, D., Wu, Q., Pu, C., Yan, W., Jung, G.P.: Approaches for service deployment. Internet Computing, IEEE 9(2), 70–80 (2005)

[20] Zhang, Y., Li, Y., Zheng, W.: Automatic software deployment using user-level virtualization for cloud-computing. Future Generation Computer Systems 29(1), 323–329 (2013).

[21] Almeida, M. et al. MODAClouds architecture - Initial version.[S.l.], May 2013. Disponível em:http://www.modaclouds.eu/wp-content/uploads/2012/09/MODACloudsD3.2.1MODACloudsArchitectureInitialVersion.pdf

[22] KDE. Umbrello The UML Modeller (Acesso em: out. 2013). Disponível em: http://umbrello.kde.org/

[23] COLLABNET INC. ArgoUML. (Acesso em: out. 2013). Disponível em:http://argouml.tigris.org/

[24] DIA DEVELOPERS. Dia Diagram Editor (Acesso em: out. 2013). Disponível em: http://dia-installer.de/

[25] MODELIOSOFT. Modelio The Open Source Modeling Environment.(Acesso: out. 2013). Disponível em:http://www.modelio.org/

[26] GAPHOR. Gaphor (Acesso em: out. 2013). Disponível em:http://gaphor.sourceforge.net/

[27] PLASTIC SOFTWARE. StarUML The Open Source UML/MDA Platform. 2013. Disponível em:http://staruml.sourceforge.net/en/

[28] TIHANYI, B. NClass (Acesso em: out. 2013). Disponível em: http://nclass.sourceforge.net/support.html

[29] VIOLET UML Editor. (Acesso: out. 2013). Disponível em: http://alexdp.free.fr/violetumleditor/page.php