PLANEJAMENTO DE AÇÕES EM SISTEMAS HIERÁRQUICOS TEMPORIZADOS USANDO GHENESYS

Arianna Z. Olivera Salmon*, Pedro M. Gonzalez Del Foyo†, José Reinaldo Silva*

*Design Lab, Departamento de Mecatrônica, Escola Politécnica, Universidade de São Paulo
Av. Professor Mello Moraes, 2231, CEP 05508-970
São Paulo/SP, Brasil

†Departamento de Engenharia Mecânica, Universidade Federal de Pernambuco
Av. Acadêmico Helio Ramos s/n, CEP 50670-901
Recife/PE, Brasil

Emails: arianna@usp.br, pedro.foyo@ufpe.br, reinaldo@usp.br

Abstract— A major issue using Automatic Planning techniques to solve real problems is the specification and modeling of requirements. Because of that, this first stage is considered at the same time, the most complex and the most strategic to achieve proper results. At this stage, errors must be detected before the implementation begins, thus avoiding the waste of time and resources. This work proposes a method for the modeling, analysis and verification of requirements, starting with a semi-formal representation in UML. Formal methods based on Petri net are used to model, analyze and verify these requirements. A case study of a real planning system of ship operations to support the logistic of oil rigs and ports will be presented and set in GHENeSys system (General Hierarchical Enhanced Net System). To well defined problems this requirements sets could be used as a basis to perform the planning avoiding the use of heuristics. Otherwise, the requirement set can be used as a start to prepare the problem to AI planning tools.

Keywords— Automatic Planning, Modeling requirements, Verification, Petri Nets.

Resumo— O uso de técnicas de Planejamento Automático para resolver problemas reais, tem entre seus principais problemas a especificação e modelagem adequada dos requisitos. Portanto, a fase inicial de eliciação, especificação e análise de requisitos, é considerada, ao mesmo tempo, a mais complexa, e a mais estratégica para obtenção de bons resultados. Erros precisam ser detectados durante esta fase inicial enquanto se antecipa ao máximo a formalização do problema. Este trabalho propõe um método para a análise, modelagem e verificação de requisitos, partindo da sua representação semi-formal em UML, e utilizando métodos formais baseados em Rede de Petri para proceder à modelagem, análise e verificação. É apresentado um estudo de caso de um sistema real de planejamento de operações de navios para atender portos e plataformas de petróleo, e ambientado no sistema GHENeSys (General Hierarchical Enhanced Net System). Para problemas bem estruturados esta base inicial de requisitos pode ser usada para fazer o planejamento usando Redes de Petri com bons resultados, dispensando a necessidade de heurísticas. Em caso contrário este conjunto próprio de requisitos pode ser usado como entrada para outras ferramentas de planejamento automático baseadas em AI (Inteligência Artificial).

Palavras-chave— Planejamento Automático, Modelagem e Análise de requisitos, Verificação, Rede de Petri.

1 Introdução

O uso de técnicas de Planejamento Automático para solucionar problemas reais, tem dividido a atenção dos pesquisadores da área de Inteligência Artificial, que em geral se dedicam aos métodos baseados em lógica, independentes do domínio de aplicação. Entretanto, para problemas reais, o conhecimento do domínio é fundamental, e estes domínios são considerados "complexos", de modo que a fase de eliciação, análise, e modelagem de requisitos é considerada crucial para o sucesso do projeto. Nesse caso, um problema que merece especial atenção é converter requisitos de uma representação semi-formal - como a UML por exemplo - para uma aboradagem formal, onde se pode fazer a análise destes requisitos e finalmente transformá-los em especificação (Liu et al., 2002).

A representação formal de destino pode variar, dependendo do domínio de aplicação, mas, para sistemas dinâmicos automatizados as Redes de Petri têm sido a mais usadas (Baresi and Pezzè, 2001). No caso da transferência de UML para redes de Petri existe ainda a possibilidade de sair de uma linguagem semi-formal diagramática, onde diferentes *features* podem ser representados por diagramas diferentes para uma representação unificada. Assim, requisitos compatíveis e sem ambiguidades devem resultar em redes consistentes e com propriedades conservativas (invariantes) bem definidas. A análise de propriedades da rede responde portanto pela análise dos requisitos, que ainda podem ser verificados no mesmo processo. O presente trabalho tem como objetivo propor uma disciplina de projeto para resolver problemas complexos de planejamento usando um método de modelagem, análise e verificação, baseado em UML e Redes de Petri (GHENeSys).

Na seção 2 descrevemos como UML e as redes de Petri estão sendo usadas na modelagem e análise de sistemas; a seção 3 descreve a metodologia de analise e modelagem de requisitos proposta. Os resultados obtidos são apresentados na seção 4, onde mostramos um estudo de caso de um sistema real de planejamento de transporte de carga por

navios para um conjunto de plataformas no mar (ambiente do pre-sal). Na seção 5 são feitas as considerações finais do trabalho.

2 UML e Redes de Petri na modelagem de sistemas

A UML oferece um conjunto de notações e diagramas para representar requisitos eliciados, observando diferentes aspectos, sem que haja uma forte interferência entre eles. Porém, a notação de requisitos em UML sofre uma dicotomia: por um lado não pode ser estritamente formal ou perderia a flexibilidade de ajuste a diferentes situações do processo de eliciação; por outro, exatamente por não poder ser formal, não pode incluir uma fase de análise (formal) de requisitos.

Portanto, para viabilizar uma fase de análise de requisitos e evitar a transferência de problemas pertinentes à fase inicial de *design* para as fases posteriores, é preciso fazer uma transferência semântica dos requisitos representados em UML para uma linguagem formal. Existe na literatura uma ampla discussão sobre a escolha da linguagem de especificação (de requisitos) mais adequada (Frappier and Habrias, 2001). Neste contexto, as redes de Petri, e em especial as redes de Petri orientadas a objeto são consideradas uma boa opção.

Vários trabalhos propõem a transferência semântica de UML para redes de Petri e suas extensões (Baresi and Pezzè, 2001), (Zhao et al., 2004), (Labbani, 2010). Muitos destes trabalhos oferecem métodos para construir redes de Petri que representem o comportamento dos sistemas partindo dos diagramas de sequência, de atividades, de estados, e até de caso de uso. Entretanto, para obter uma boa transferência semântica dos requisitos deve-se levar em consideração tanto a estrutura estática e dinâmica dos diagramas, como as relações entre eles. Em outras palavras deve-se capturar o funcionamento integrado entre as partes.

3 A Disciplina de Projeto UML-GHENeSys

Neste trabalho estamos propondo construir uma rede de P/T (place/transition) hierárquica a partir dos modelos que integram diagramas com vistas estática e dinâmica, possibilitando assim a verificação formal destes modelos. Assim, serão utilizadas extensões das redes clássicas (presentes na norma ISO/IEC 15.909) das RdP, especialmente a estrutura hierárquica presente no ambiente GHENeSys (General Hierarchical Enhanced Net System)(del Foyo, 2009). Este ambiente está sendo preparado para cobrir todos os itens da norma, incluindo as extensões (ISO/IEC 15.909-3), como hierarquia, pseudo-boxes - associados aos *gates* e exclusivamente definidos para o ambiente GHENeSys - e a extensão temporal. Mesmo incluindo estas extensões, o GHENeSys provê uma equação de estado (para as redes P/T) formalmente definida (del Foyo, 2009), o que posibilita a analise das propriedades da rede.

A disciplina de projeto apresentada neste trabalho, consiste no uso da linguagem UML para eliciar e representar os requisitos, numa primeira etapa, e em seguida usar o ambiente GHENeSys para prover uma representação formal. Proporemos a análise de invariantes como parte do processo de análise e validação de requisitos, e também na fase de verificação.

Embora a UML possua um alto poder de expressão gráfica, existem propriedades e restrições de sistemas que são muito complexas ou impossíveis de serem expressas adequadamente em um diagrama visual (*visual diagram*). Mesmo os mecanismos de extensão da UML: estereótipos, valores etiquetados e restrições pré-definidas; podem ser insuficientes para compor uma especificação de requisitos completa[1]. Assim, será usada a OCL para formular alguns dos requisitos (restrições), especialmente aqueles que representam invariantes. Uma vez feita a modelagem em UML, estes requisitos serão transformados em uma rede GHENeSys, e posteriormente verificados formalmente, usando o método de análise de invariantes (Salmon et al., 2014).A disciplina de projeto para a fase inicial é descrita a seguir:

1. Representar os requisitos do sistemas usando os diagramas de classes, de estados e de tempo da UML.
2. Definir restrições para expressar os requisitos do sistema que sejam invariantes, usando a linguagem OCL.
3. Transformar as restrições em OCL em um conjunto de inequações que representem os invariantes de lugar de uma rede de Petri.
4. Transformar a representação UML em uma rede GHENeSys através de um método desenvolvido pelos autores deste trabalho, sobre a proposta de (Baresi and Pezzè, 2001). As figuras 1, 2 mostram o mapeamento dos diagramas UML na rede GHENeSys.
5. Verificar requisitos detectando inconsistências e/ou usando os invariantes da rede de Petri.

4 Resultados

A seguir mostramos os resultados obtidos no estudo de caso de um sistema real de planejamento

[1] Neste trabalho, chamaremos especificação de requisitos *completa* àquela que permite compor o arquétipo de um artefato e não apenas aspectos dele.

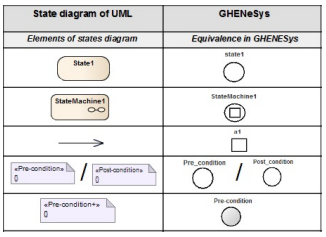

Figura 1: Mapeamento dos elementos do diagrma de estados de UML e os elementos da rede GHENeSys

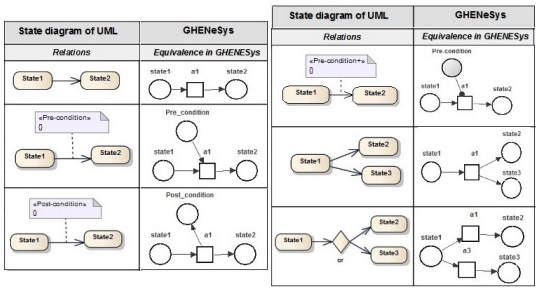

Figura 2: Mapeamento das relações entre elementos do diagrama de estados de UML e a rede GHENeSys

de operações de navios em plataformas e portos de petróleo.

4.1 Descrição do estudo de caso

Genericamente, o problema que se apresenta como estudo de caso pode ser descrito como a necessidade de fornecer o transporte de mercadorias e de ferramentas entre portos e plataformas de exploração e/ou produção no oceano.

Para simplificar o problema, optou-se por colocar duas plataformas na faixa de Rio de Janeiro, e duas no Santos. Consideraremos três embarcações, ou seja, grandes barcos que podem transportar várias toneladas de carga. O que realmente faz a diferença para o problema é que isso leva algum tempo para carregar/descargar a carga.

Dividimos estas faixas em duas partes: de Rio de Janeiro e Santos, respectivamente. Cada faixa tem um porto (porto *P1* no Rio de Janeiro e porto *P2* em Santos), onde as atividades de carrega/descarga ocorrem para apoiar a extração de petróleo em águas profundas (pré-sal). Ambas faixas contem um conjunto de plataformas: duas plataformas (*F1, F2*) na faixa de Rio de Janeiro, e duas (G1,G2) na faixa de Santos. Existem também áreas de espera(uma área de espera em cada faixa): a do Rio de Janeiro (chamado *A1*) está localizada a 120 km (distância radial) do porto *P1* e o de Santos (chamado de *A2*) está localizado a 100 km do porto *P2* onde ficam ancoradas as embarcações para não ocupar lugar nos respectivos portos.

4.2 Especificações do sistema

Dado um conjunto de *itens* de carga, o problema consiste em encontrar um plano viável que garanta a sua entrega, respeitando as limitações e exigências da capacidade dos navios. O objetivo é minimizar a quantidade total de combustível consumido, o tamanho das filas de espera nos portos, o número de navios utilizados, e o custo de ancoragem. Um objetivo especial é reduzir a zero a fila de espera para ancoragem junto às plataformas.

4.3 Restrições do sistema a ser consideradas neste trabalho

- Somente um navio, por vez, pode ancorar na plataforma de reabastecimento.
- Não é possível carregar e descarregar o navio ao mesmo tempo: primeiro descarrega e depois carrega.
- O reabastecimento pode ser feito durante a ação de carga ou descarga.
- O tempo de ancoragem e desancoragem nos portos é de até uma hora, e nas plataformas de até 30 minutos.
- O tempo para carregar e descarregar os navios tanto nos portos como nas plataformas é até 5 horas.
- O reabastecimento de combustível nos portos pode demorar até 3 horas, e nas plataformas até 5 horas.

Consideraremos como posição inicial dos navios as áreas de espera A1 e A2. F2 e G2 são as únicas plataformas capazes de realizar o reabastecimento, além dos portos.

4.4 Modelagem do sistema de planejamento

O primeiro passo na modelagem do sistema descrito é representar os requisitos usando UML. Primeiramente se representam os principais objetos do sistema e suas relações através do diagrama de classes da UML.
Partindo do diagrama de classes da UML são construídos os diagrama de estados. A figura 3 mostra a hierarquia destes diagramas de estados.
As restrições de tempo do sistema são representadas através dos diagramas de tempo. Para este trabalho somente consideraremos as restrições de tempo correspondente as ações de ancoragem/desancoragem, carregamento e descarregamentos dos navios, tanto nos portos como nas plataformas. A figura 4 mostram as restrições de tempo para o navio nos portos e nas plataformas respectivamente.

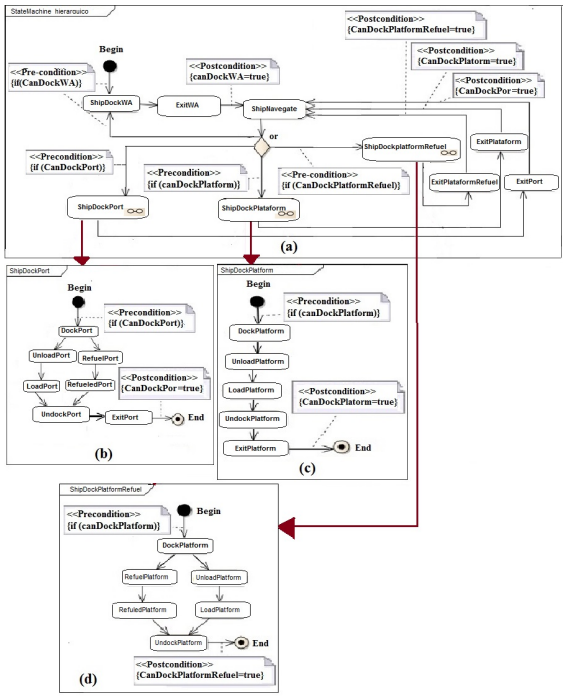

Figura 3: Hirarquía dos diagrams de estados.(a) Diagrama de estado Geral, (b) Diagrama de estado correspondentes aos portos, (c) Diagrama de estados correspondentes as plataformas, (d) Diagrama de estados correspondentes as plataformas de reabastecimento

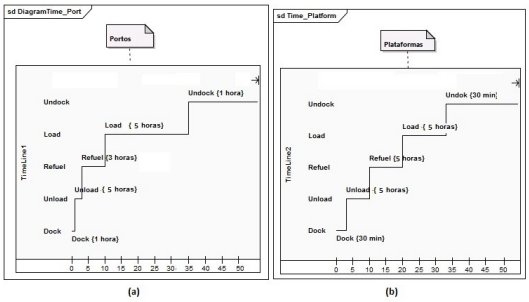

Figura 4: Diagrama de tempo para operações do navio em: ***(a)Portos***, ***(b)Plataformas***.

4.4.1 Transformando os diagramas UML em rede de Petri

Uma vez feita a modelagem em UML, os diagramas de estados são transformados numa rede GHENeSys como foi descrito na seção 3.

As ações dos navios nos portos e nas plataformas foram representadas através de três subdiagramas de estados: o primeiro representa as ações do navio nos portos, enquanto as operações das plataformas foram representadas em outros dois diagramas de estados: um para as plataformas onde não é possível reabastecer os navios, e outro para representar o movimento relativo de navios e plataformas onde é possível o reabastecimento. A figura 5 mostra a transformação do modelo hierárquico de diagramas de estados, mostrado na figura 4, em uma rede hierárquica GHENeSys. Porém esta rede não representa o problema real, uma vez que o sistema de planejamento que estamos modelando possui 3 navios, 2 portos, e 4 plataformas, duas delas com possibilidade de reabastecimento.

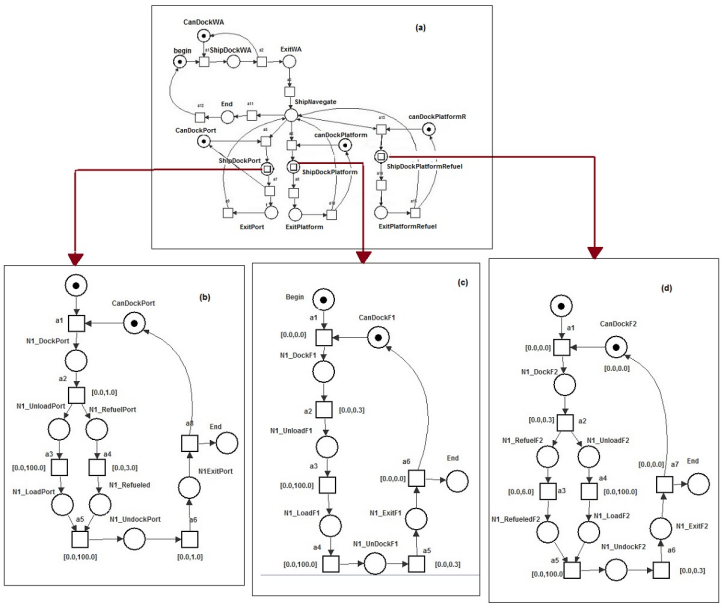

Figura 5: Modelagem hierárquica do sistema na rede GHENeSys. (a) Rede geral do sistema, (b) Rede correspondentes aos portos, (c) Rede correspondentes as plataformas, (d) Rede correspondentes as plataformas de reabastecimento

Sendo assim, precisamos modelar todas as instâncias do sistema. Numa rede de Petri, as instâncias são modeladas por marcas. Para o sistema que estamos modelando as marcas representam a presença ou não de um navio em cada estado (lugar da rede). Porém, neste sistema de planejamento os navios tem características diferentes, e portanto não seria adequado colocar varias marcas em um mesmo lugar para representar diferentes navios. Consequentemente, para cada estado do diagrama de estados principal do sistema figura 3 (a) devem ser representados por três lugares diferentes na rede GHENeSys, que representarão os três navios. No caso dos estados que representam as áreas de espera, e nos portos teremos seis lugares diferentes para cada um, devido a que temos duas áreas de espera e três navios. Para as plataformas serão considerados doze lugares. A seguir, na tabela 1 se mostra o mapeamento de alguns dos estados do diagrama de estados geral figura 3(a) e os lugares da rede mostrada na figura 5(a).

A figura 6 mostra a rede hierárquica do sistema, já considerando todas suas instâncias.

Diagrama de estados geral		
Estado	*Lugar*	*Descrição*
ShipDockWA	$N_iDockWA1$, $N_iDockWA2$ $i = 1...3$	Navio (i) ancorado na área de espera 1 Navio (i) ancorado na área de espera 2
ShipDockPort	$N_iDockP1$, $N_iDockP2$ $i = 1...3$	Navio (i) ancorado no porto 1 Navio (i) ancorado no porto 2
ShipDockPlatform	$N_iDockF1$, $N_iDockG1$ $i = 1...3$	Navio (i) ancorado na plataforma F1 Navio (i) ancorado na plataforma G1
ShipDockPlatform Refuel	$N_iDockF2$, $N_iDockG2$ $i = 1...3$	Navio (i) ancorado na plataforma F2 Navio (i) ancorado na plataforma G2

Tabela 1: Equivalência entre os estados no diagrama de estados e os lugares na rede GHENeSys.

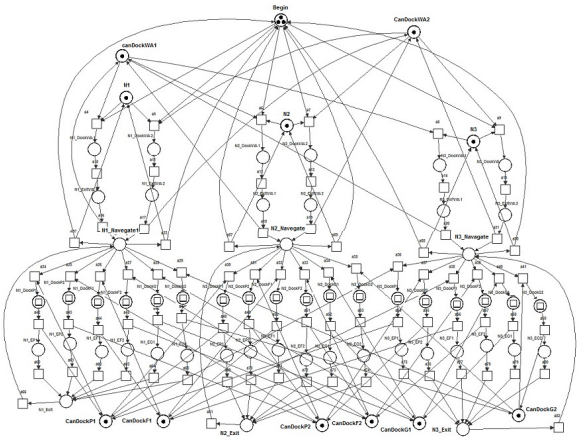

Figura 6: Rede GHENeSys do sistema geral.

4.5 Verificação dos modelos

Partindo da informação representada no diagrama de classes e das restrições formuladas em OCL, se definem alguns requisitos funcionais do sistema que precisam ser verificados. Os requisitos serão verificados usando os invariantes de lugar das redes de Petri.

A tabela 2 mostra as descrições em OCL correspondente as operações dos navios nos portos, e sua interpretação em termos de inequações que representam os invariantes de lugar.

A equação (1) da tabela 2 significa que enquanto os navios estão fazendo a operação de carregamento nos portos, não podem fazer o descarregamento. A equação (2) significa que enquanto o navio faz operações de carga ou descarga nos portos, é possível reabastecer o tanque.

A tabela 3 descreve os requisitos que devem ser verificados nas operações dos navios nas plataformas onde não é possível reabastecer. Neste caso só se verifica que as operações de carga e descarga não se realizem de forma simultânea.

Id	Especificações em OCL	**Invariantes**
1	(self.Load()) **implies** **not** ((self.Unload())	$M(N_iLoadP_j)+$ $M(N_iUnloadP_j) \leq 1$ $i = 1...3, j = 1...2$
2	if (self.Load() *or* self.Unload()) **implies** self.Refuel()	$M(N_iLoadP_j)+$ $M(N_iUnloadP_j) \leq 1$ $i = 1...3, j = 1...2$

Tabela 2: Especificação descritas em OCL correspondente as operações dos navios nos portos.

Id	Especificações em OCL	**Invariantes**
1	(self.Load()) **implies** **not** ((self.Unload())	$M(N_iLoadF1)+$ $M(N_iUnloadF1) \leq 1$
2		$M(N_iLoadG1)+$ $M(N_iUnloadG2) \leq 1$

Tabela 3: Especificação descritas em OCL correspondente as operações dos navios nas plataformas.

A figura 7 mostra alguns dos invariantes de lugar, corresponde à rede dos Portos (figura 5 (b)), e à rede das Plataformas (figura 5 (c)), obtidos pelo sistema GHENeSys. A equação 1 da tabela 2 se verifica com a invariante numero 1 da figura 7*(a)*; e a equação 2 se verifica com o invariante numero 5. As equações da tabela 3 podem ser verificadas com a invariante numero 1 da figura 7(b). A figura 7*(b)* mostra o calculo dos invariante na rede correspondente a plataforma *F1*, mas o calculo dos invariantes na rede da plataforma *G1* é equivalente.

	Lugares	Invariantes de lugar 1	2	3	4	5
1	Begin	0	0	1	1	1
2	N1_DockPort	1	1	1	1	2
3	N1_UnloadPort	1	0	1	0	1
4	N1_RefuelPort	0	1	0	1	1
5	N1_LoadPort	1	0	1	0	1
6	N1_Refueled	0	1	0	1	1
7	N1_UndockPort	1	1	1	1	2
8	CanDockPort	1	1	0	0	1
9	N1ExitPort	1	1	1	1	2
10	End	0	0	1	1	1

(a)

	Lugares	Invariantes 1	2
1	Begin	1	0
2	N1_DockF1	1	1
3	N1_UnloadF1	1	1
4	N1_LoadF1	1	1
5	N1_UnDockF1	1	1
6	N1_ExitF1	1	1
7	End	1	0
8	CanDockF1	0	1

(b)

Figura 7: **(a)**: Invariantes de lugar da rede que modela as ações nos portos (figura 5 (b)). **(b)**: Invariantes de lugar da rede que modela as ações nas plataformas (figura 5 (c)).

Além dos requisitos formulados em OCL e representados no diagrama de classes, existem outros requisitos do sistema que podem ser verificados usando o cálculo dos invariantes. A tabela 4 descreve os invariantes que devem ser verificados no sistema geral, representado na rede da figura 6.

As equações da tabela 4 representam os seguintes requisitos do sistema: a equação 1 significa que os navios podem estar em apenas um dos seguintes lugares: áreas de espera, portos, plataformas, ou navegando; a equação 2, 3 e 4 significa nas plataformas só pode ter um navio de cada vez.

A figura 8 mostra os resultados do calculo dos invariantes de lugar, obtidos pelo sistema GHENeSys, na rede correspondente ao sistema geral, mostrada na figura 6. A equação numero 1 da tabela 4 se verifica com o invariante 4 da figura

Id	Invariantes
1	$M(N_i DockWA1) + M(N_i DockWA2) +$ $M(N_i DockP1) + M(N_i DockP2) +$ m $M(N_i DockF1) + M(N_i DockF2) +$ $M(N_i DockG1) + M(N_i 1DockG2) +$ $M(N_i Navegate)$, $i = 1...3$
2	$M(N_1 DockF1) + M(N_2 DockWF1) +$ $M(N_3 DockF1)$
3	$M(N_1 DockF2) + M(N_2 DockWF2) +$ $M(N_3 DockF2)$
4	$M(N_1 DockG1) + M(N_2 DockWG1) +$ $M(N_3 DockG1)$

Tabela 4: Invariantes do sistema geral.

8 (a cor laranja pertence ao navio N1, a cor azul ao navio N2, e a cor vermelha ao navio N3). As restantes equações (2, 3, 4) são verificadas pelas invariantes (1, 2, 3) da figura 8, respectivamente.

		Invariantes de lugar			
	Lugares	1	2	3	4
1	canDockWA1	1	1	1	0
2	CanDockWA2	1	1	1	0
3	N1_DockWA1	1	1	1	1
4	N1_DockWA2	1	1	1	1
5	N2_DockWA1	1	1	1	1
6	N2_DockWA2	1	1	1	1
7	N3_DockWA1	1	1	1	1
8	N3_DockWA2	1	1	1	1
9	N1_EstWA1	1	1	1	1
10	N1_EstWA2	1	1	1	1
11	N2_EstWA1	1	1	1	1
12	N2_EstWA2	1	1	1	1
13	N3_EstWA1	1	1	1	1
14	N3_EstWA2	1	1	1	1
15	N1_Navegate	0	0	0	1
16	N2_Navegate	0	0	0	1
17	N3_Navegate	0	0	0	1
18	CanDockP1	1	1	1	0
19	CanDockP2	1	1	1	0
20	CanDockF1	1	1	1	0
21	CanDockF2	0	1	1	0
22	CanDockG1	0	0	1	0
23	CanDockG2	0	0	0	0
24	N1_DockP1	1	1	1	1
25	N1_DockP2	1	1	1	1
26	N1_DockF1	1	1	1	1
27	N1_DockF2	0	1	1	1
28	N1_DockG1	0	0	1	1
29	N1_DockG2	0	0	0	1
30	N2_DockP1	1	1	1	1
31	N2_DockP2	1	1	1	1
32	N2_DockF1	1	1	1	1
33	N2_DockF2	0	1	1	1
34	N2_DockG1	0	0	1	1
35	N2_DockG2	0	0	0	1
36	N3_DockP1	1	1	1	1
37	N3_DockP2	1	1	1	1
38	N3_DockF1	1	1	1	1
39	N3_DockF2	0	1	1	1
40	N3_DockG2	0	0	0	1
41	N3_DockG1	0	0	1	1

	Lugares	Invariantes de lugar 1	2	3	4
42	N1_EP1	0	0	0	1
43	N1_EP2	0	0	0	1
44	N1_EF1	0	0	0	1
45	N1_EF2	0	0	0	1
46	N1_EG1	0	0	0	1
47	N1_EG2	0	0	0	1
48	N1_Exit	0	0	0	1
49	N2_EP1	0	0	0	1
50	N2_EP2	0	0	0	1
51	N2_EF1	0	0	0	1
52	N2_EF2	0	0	0	1
53	N2_EG1	0	0	0	1
54	N2_EG2	0	0	0	1
55	N3_EP1	0	0	0	1
56	N3_EP2	0	0	0	1
57	N3_EF1	0	0	0	1
58	N3_EF2	0	0	0	1
59	N3_EG1	0	0	0	1
60	N3_EG2	0	0	0	1
61	N2_Exit	0	0	0	1
62	N3_Exit	0	0	0	1
63	Begin	0	0	0	0
64	N1	0	0	0	1
65	N2	0	0	0	1
66	N3	0	0	0	1

Figura 8: Invariantes de lugar da rede principal (figura 6).

5 Conclusões

Neste trabalho foi apresentado um método de *desing* para sistemas reais baseado em UML e redes de Petri. O método proposto usa os diagramas de classes, estados e tempo na representação inicial dos requisitos. Uma vez sintetizado o modelo em uma, ou várias redes GHENeSys, passamos à fase de verificação. Para este trabalho, a proposta foi usar a análise dos invariantes da rede de Petri, na verificação formal dos requisitos funcionais e estruturais do sistemas. Assim uma vez construída a rede GHENeSys, os invariantes devem ser calculados para demonstrar a veracidade ou não das propriedades verificadas.

O método proposto neste artigo preenche uma lacuna da maioria dos métodos similares que é ter um procedimento de análise dos requisitos baseado em uma representação formal. Permite ainda uma validação antecipada antes da formalização das especificações, tornando a fase de *desing* e provimento de soluções mais consistente, baseada em dados confiáveis. A escalabilidade é ainda um problema, mesmo usando uma rede hierárquica, dado que foi usada uma rede P/T clássica. Entretanto o fato de que os invariantes (a base do método) foram inseridos na fase inicial (usando OCL) torna factível a introdução de redes de alto nível. Neste caso uma proposta mais arrojada de análise de invariantes deve ser também tratada.

Finalmente, o uso de um sistema temporizado abre a possibilidade de não só fazer o planejamento do sistema de transporte mas o escalonamento do sistema, sincronizando as operações de ancoragem, caga e descarga e verificando o problema com *model checking* baseado em redes de Petri.

Agradecimentos

A autora Arianna Z. Olivera Salmon tem suporte através de bolsas de estudo financiada pela Agência Nacional do Petróleo, Gás Natural e Biocombustíveis- ANP -, da Financiadora de Estudos e Projetos - FINEP- e do Ministério da Ciência, Tecnologia e Inovação -MCTI por meio do Programa de Recursos Humanos da ANP para o Setor Petróleo e Gás- PRH-ANP/MCTI.

Referências

Baresi, L. and Pezzè, M. (2001). Improving uml with petri nets, *Electronic Notes in Theoretical Computer Science* **44**: 107–119.

del Foyo, P. M. G. (2009). *Verificação Formal de Sistemas Discretos Distribuídos*, PhD thesis, Escola Politécnica da Universidade de São Paulo.

Frappier, M. and Habrias, H. (2001). *Software specification methods: an overview using a case study*, Springer-Verlag.

Labbani, O. (2010). A uml and colored petri nets integrated modeling and analysis approach using graph transformation, *Journal of Object Technology* **9**(4).

Liu, L., Yu, E. and Mylopoulos, J. (2002). Analyzing security requirements as relationships among strategic actors, *Submitted to the Symposium on Requirements Engineering for Information Security (SREIS'02), Raleigh, North Carolina*.

Salmon, A. Z. O., del Foyo, P. M. and Silva, J. R. (2014). Verification of automated systems using invariants.

Zhao, Y., Fan, Y., Bai, X., Wang, Y., Cai, H. and Ding, W. (2004). Towards formal verification of UML diagrams based on graph transformation, *Proceedings of the IEEE International Conference on E-Commerce Technology for Dynamic E-Business*, IEEE Computer Society.