Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática – CEEI
Departamento de Sistemas e Computação – DSC

Operação via Regras para o escalonamento de bombas hidráulicas

Thiago Emmanuel Pereira da Cunha Silva

Resumo

O escalonamento de bombas numa malha de produção de pétroleo é uma decisão que pode impactar em toda o processo produtivo da empresa, seja na grande diferença de custo entre as soluções ou em resultados que mudem o estado físico da malha, como o aumento de pressão e vazão na rede de distribuição, esvaziamento/transbordo de tanque, que podem trazer prejuízos ambientais e financeiros.

Modelos computacionais podem ser sugeridos para solucionar este problema como os algoritmos evolucionários etc. Talvez a abordagem evolucionária mais típica seja os Algoritmos genéticos.

Outra opção de modelagem é a operação baseada em regra, um modelo que tenta reproduzir o conhecimento de um agente em determinado processo.

Neste trabalho mostramos uma especificação de operação via regras desenvolvida pela equipe de engenharia do projeto *SmartPumping*,

Introdução

O problema do escalonamento de bombas, ou seja, a definição de uma programação de liga-desliga pode ser mapeado atráves de algoritmos genéticos, onde as possíveis soluções considerada pelo algoritmo (os cromossomos) teriam um representação inteira binária que traduzem o estado atual de uma bomba, a ordem de cada bit dentro de um cromossomo indicam

a evolução no tempo do estado da bomba representada.

Muitos autores destacam as vantagens do algoritmo genético em relação a outros métodos de otimização, entre outros motivos:

- Realizam buscas simultâneas em várias regiões do espaço de busca, pois trabalham com uma população e não com um único ponto;
- Utilizam apenas informações da função objetivo, não requerendo o uso de derivadas ou outro conhecimento auxiliar;
- Funcionam tanto com parâmetros contínuos como discretos ou uma combinação deles;
- São fáceis de serem implementados e adaptam-se bem a computadores paralelos;

Entreto estes algoritmos possuem algumas deficiências como por exemplo a convergência e a dificuldade de refinar soluções próximas de um ótimo global.

Uma opção aos algoritmos genéticos é o modelo de operação baseada em regras, este modelo apresenta desempenho notavelmente superior ao modelo previamente explicado entretando sua soluções geralmente apresentam caracter sub-ótimo.

Este trabalho descreve a implementação de um modelo de operação baseado em regras para o escalonamento do conjunto de bombas de uma rede de escoamento de petróleo especificado pelo equipe de engenharia do projeto SmartPumping e uma descrição de uma alternativa de operação baseada num sistema fuzzy.

A implementação foi feita usando o framework Drools, sua escolha baseou-se na familiaridade do autor com a linguagem de programação Java

bem como a facilidade do uso de alguns aspectos do framework por pessoas de áreas não diretamente ligadas a computação. Entre estes aspectos está o uso de uma linguagem mais próxima da natural, facilitando a descrição de regras do tipo SE<> ENTAO<> por exemplo.

Lista de Tabela

Lista de Figuras

2. Regras de Operação

O conjunto de regras de atuação individual para operação da rede possui duas características principais: 1) objetiva manter um padrão de vazão de chegada na estação receptora, definido pelo usuário; 2) opera individualmente em cada estação, ligando ou desligando bombas de acordo com o nível do tanque e a diferença percentual entre o padrão de vazão desejado e a vazão monitorada na estação receptora.

Essa abordagem utiliza o erro percentual entre a vazão desejada para cada intervalo de atuação e a vazão obtida com a configuração das bombas no intervalo de atuação anterior e utiliza o nível dos tanques produtores para determinar o percentual de bombas que devem ser ligadas ou desligadas para que a vazão atinja o valor desejado.

O erro percentual é classificado em nove faixas, denominadas: erro negativo muito grande (NMG), erro negativo grande (NG), erro negativo médio (NM), erro negativo pequeno (NP), faixa de erro aceitável (aceitável), erro positivo pequeno (PP), erro positivo médio (PM), erro positivo grande (PG) e erro positivo muito grande (PMG). A dimensão de cada faixa de erro é definida pelo usuário.

Para aplicação desse conjunto de regras, cada tanque produtor deve ter seu volume útil dividido em cinco faixas: Nível muito baixo - MB; Nível baixo – NB; Nível médio – NM; Nível alto – NA e Nível muito alto – MA, conforme a Figura 2. Essas faixas são caracterizadas por porcentagens, definidas pelo usuário, do volume útil dos tanques produtores, ou seja, do

nível de controle mínimo até o nível de controle máximo:

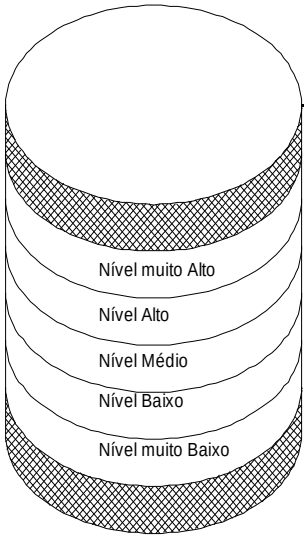


Figura 2 – Faixas de volume útil do tanque produtor para regras de atuação integradas

Desse modo, constituem dados de entrada para a aplicação dessas regras:

- Vazão de entrada desejada para a estação receptora para cada intervalo de atuação t : QD_t
- As faixas dos níveis dos tanques produtores;
- As faixas do erro percentual.

E constituem dados a serem monitorados do sistema a cada intervalo de

atuação:

- Vazão de entrada na estação receptora: QM_t ;
- Níveis dos tanques produtores: $N1_t$, $N2_t$, ... , Nn_t ;
- Estado das bombas do sistema.

A faixa denominada muito baixa – MB determina que todas as bombas da estação sejam desligadas, independente do erro entre a vazão desejada e a vazão monitorada. Da mesma forma que, quando o nível do tanque estiver contido na faixa muito alta – MA, a determinação é que todas as bombas conectadas a esse tanque sejam ligadas, independente do erro.

Nas demais faixas de nível – Alto, Médio e Baixo – a determinação das bombas a serem ligadas ou desligadas é realizada considerando-se o erro entre a vazão desejada do intervalo atual e a vazão monitorada do intervalo anterior, ou seja, a vazão que provavelmente se obterá caso se mantenha a programação das bombas. Quando o erro estiver na faixa definida como aceitável, a recomendação das regras sempre será para manter a programação das bombas.

$$Erro_t = \frac{QD_t \; QM_{t\;1}}{QD_t} \cdot 100$$

onde:

$Erro_t$ é o erro percentual entre a vazão desejada e a vazão monitorada;

QD_t é a vazão desejada para o intervalo de atuação t ;

$QM_{t\;1}$ é a vazão monitorada no intervalo de atuação anterior.

A cada intervalo de atuação, uma vez determinada a magnitude do erro e observada a faixa de nível em que cada tanque produtor se encontra,

aplicam-se as recomendações da Tabela 1 a cada estação individualmente, definindo assim, a nova configuração das bombas do intervalo.

Tabela 2 - Quadro de regras de atuação individual

<table>
<tr><th>Erro (%)
Nível do tanque</th><td>NMG</td><td>NG</td><td>NM</td><td>NP</td><td>Aceitável</td><td>PP</td><td>PM</td><td>PG</td><td>PMG</td></tr>
<tr><td>MA</td><td colspan="9">Ligar todas as bombas da estação</td></tr>
<tr><td>A</td><td rowspan="4">Desligar 50%*</td><td>Desligar 30%*</td><td>Desligar 20%*</td><td>Desligar 20%*</td><td rowspan="4">Manter a programação de bombas</td><td>Ligar 30%**</td><td>Ligar 40%**</td><td>Ligar 40%**</td><td rowspan="4">Ligar 50%**</td></tr>
<tr><td>M</td></tr>
<tr><td rowspan="2">B</td><td>Desligar 40%*</td><td>Desligar 30%*</td><td>Desligar 20%*</td><td>Ligar 20%**</td><td>Ligar 30%**</td><td>Ligar 40%**</td></tr>
<tr><td>Desligar 40%*</td><td>Desligar 40%*</td><td>Desligar 30%*</td><td>Ligar 20%**</td><td>Ligar 20%**</td><td>Ligar 30%**</td></tr>
<tr><td>MB</td><td colspan="9">Desligar todas as bombas da estação</td></tr>
</table>

* Considerando as bombas da estação que estão ligadas.

** Considerando as bombas da estação que estão desligadas.

2.1 Implementação

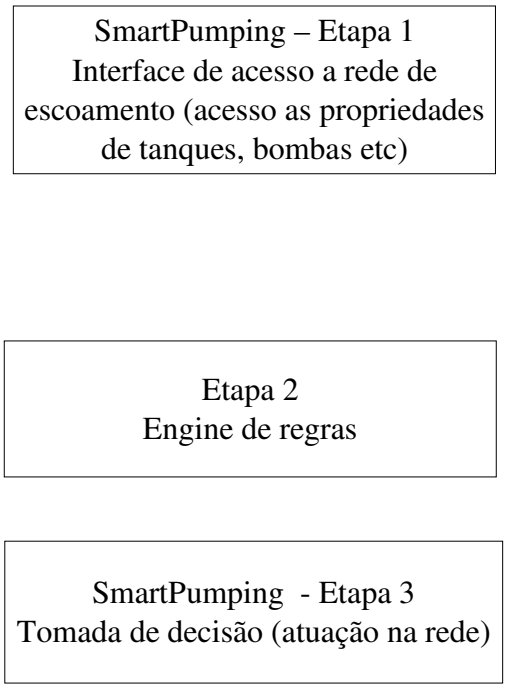
Sistemas baseados em regras assim como sistemas fuzzy traduzem para um modelo as decisões e a experiência de operadores humanos, que muitas vezes pode ser imprecisa, qualitativas e vagas. Durante o processo de desenvolvimento de sistemas dessa qualidade é precisso extrair a definição de

regras de um especialista no processo que dificilmente não o mesmo que está desenvolvendo o sistemas, assim é comum ver equipes mistas de engenheiros de software e outras categorias de profissionais, particulamente no SmartPumping existe uma equipe de Engenheiros Civis que trabalha em conjunto com engenheiros de software. Durante a formalização do modelo de regras tão mais produtivo fica o trabalho quanto as equipes consigam estabelecer meios de descrever o modelo que seja acessíveis para a equipe inteira. Um modelo descrito numa linguagem de programação de propósito embora seja mais natural para a equipe de software não ajuda a equipe de engenheiros, de certo modo engessa o processo porque os artefatos gerados não são compreendidos pela equipe que os especificou e qualquer mudança na estrutura das regras requer um novo esforço de especificação, programação e testes.

Para solucionar estes problemas resolveu-se adotar a solução open

source *Drools* fornecidade pelo grupo *Jboss*.

Usando esta solução podemos definir num arquivo texto separado e usando uma linguagem mais abstrata para a definição das regras do sistema, diminuindo a depedência com a estrutura do resto do software que é escrito na linguagem de programação *Java*, a avaliação das regras é feita por uma *engine* fornecida pelo *Drools.* Em resumo o sistema foi estruturado da seguinte forma:



É necessária a criação de interfaces bem definidas na comunicação entre a engine de regras e o resto do sistema, por exemplo no nosso caso, na comunicação entre a etapa 1 e 2 definimos uma interface para obtenção de informações do sistema como por exemplo a altura e níveis dos tanques, veja

a codificação desta regra usando o *Drools*:

```
rule "Equals Rule"
        when
                inf : Information(error == 0)
                 t : Tanque()
                man : ActionManager()
        then {
                String saida = null;
                saida = nivel(t.getNivel(), t.getAltura());

                if(saida.equals("POS_ALTO")){

                      //liga todas as bombas da estacao
                       man.atuaBombas(true, 1);

                }else if(saida.equals("NEG_ALTO")){

                      //desliga todas as bombas da estacao
                       man.atuaBombas(false, 1);
                }else {
                       //do nothing! Mantém a programação
                }
                System.out.println( "Vazão desejada é igual a monitorada" );//logar
        }
```

Os métodos getNivel() e getAltura() foram definidos em *Java* na descrição da interface para um Objeto do tipo tanque. A engine do *Drools* se encarrega de encarrega de converter este comando para uma chamada em um objeto *Java* e manipular o resultado da chamada.

Outro exemplo de interface entre os dois sistemas é a tomada de decisão a partir das regras, veja a chamada ***man.atuaBombas(true, 1);***

Para isso codificou-se um módulo no SmartPumping que altera o estado

das bombas da rede de escoamento, e a chamada para este módulo pode ser feita dentro das definições das regras.

A linguagem de definição de regras do *Drools* também permite a definições de funções por exemplo foi usada esta função para classificar o estado de um tanque em relação ao nível da sua coluna de fluído, esta relação linguistica entre a altura do tanque e o nível da coluna de fluído torna mais fácil e natural o entendimento das regras, por exemplo o nível é tratado com váriaveis do tipo ***NEG_BAIXO, POS_BAIXO*** etc:

```
/**
*       Funcao que define a relação entre nivel da coluna de fluido e
*       altura do tanque.
*/
function  String nivel(double nivel, altura){

       if(nivel < 0.6*altura && nivel >= 0.4*altura){
                return "ZERO";
        }

        else if(nivel < 0.4*altura && nivel >= 0.1*altura){
                return "NEG_BAIXO";
        }

        else if(nivel < 0.1*altura){
                return "NEG_ALTO";
        }

        else if(nivel < 0.9*altura && nivel >= 0.6*altura){
                return "POS_BAIXO";
        }

        return "POS_ALTO";
}
```

Estes dois exemplo mostrados acima, principalmente o primeiro, mostram as vantagens do uso de *Drools*, vejamos, suponha que após experimentos a equipe de engenharia decida que um tanque quando tiver seu nível acima do normal não deve ter suas bombas completamentes desligadas e

sim apenas metade delas, então o próprio engenheiro mudaria uma única linha de código na definição da sua regra e o sistema já estaria e conformidade com a nova especificação:

```
if(saida.equals("POS_ALTO")){
      //liga todas as bombas da estacao          //liga metade das bombas da estacao
   man.atuaBombas(true, 1);  ----->  man.atuaBombas(true, 0.5)
}
```

3. Operação Fuzzy

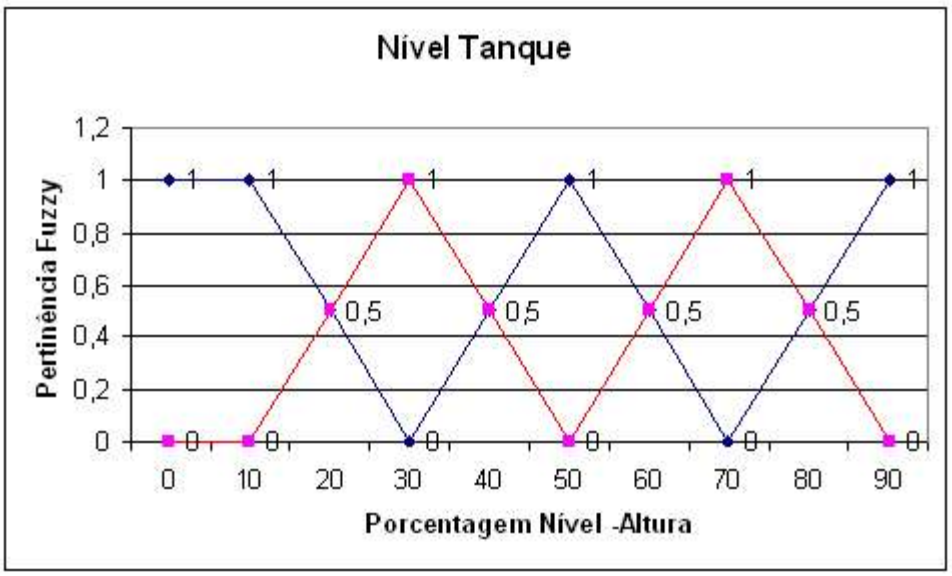
Um modelo de operação inspirado em regras inerentemente baseia-se em uma aproximação da realidade criada muitas vezes a partir do conhecimento de um especialista. Uma técnica comum e que traz um aumento da robustez do modelo é a modelagem usando um Sistemas Difuso de Regras (SDR).

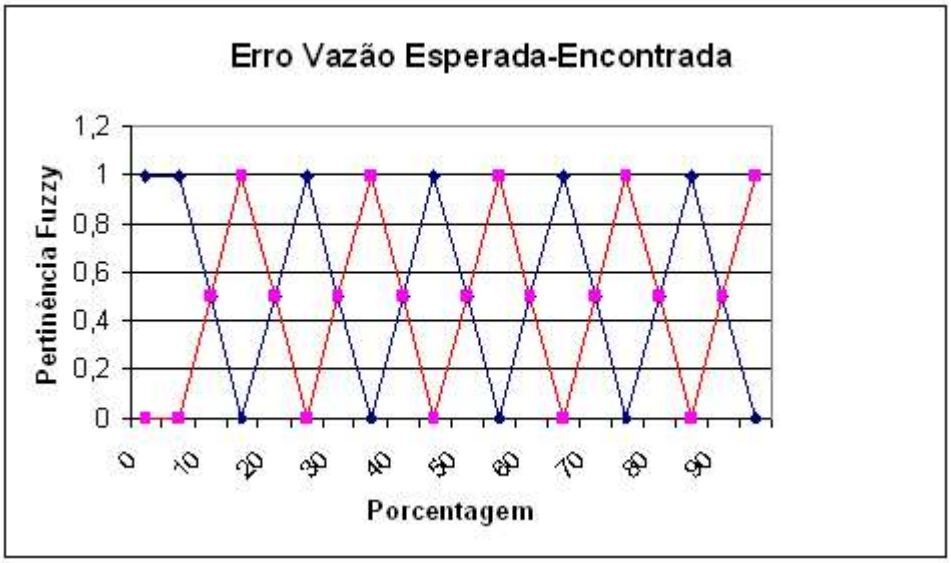
Assim como os Sistemas Especialistas (com os quais guardam grandes semelhanças) os SDR quardam e processam o conhecimento em regras do tipo: SE A=a E/OU B=b ENTÃo C=c, onde A, B e C são variáveis do processo e a, b e c valores assumidos por estas variáveis.

Estas variáveis nos SDR podem ser linguisticas ou números difusos. Voltando ao nosso sistemas as duas variáveis de entrada (Erro percentual entre Vazão desejada e Vazão encontrada, percentual entre nível da coluna de tanque e altura do tanque) tornaram-se variáveis fuzzy. A escolha da segunda

variável como fuzzy é natural por exemplo um valor de nível que estão no limiar entre médio e alto deveria levar à consequências que consideram os dois patamares porque é um valor que pode-se dizer não totalmente alto nem totalmente médio. Quanto a primeira variável à princípio não a vemos como fuzzy basicamente porque é um escolha do usuário na hora da execução do sistema, mas resolve-se torná-la fuzzy para acrescentar robustez ao sistemas perante as escolhas do usuário, por exemplo qual o exato significado físico da da adoção de vazão que traga um erro 49% de outra que traga 51,5% ?

O trabalho em fuzzyficar o sistema em resumo baseou-se em criar funções de pertinência para as variáveis já definidas na seção anterior, por exemplo níveis de tanques como 51,01% e 49,% que antes eram totalmente separados em duas categorias agora tem uma probabilidade associada de pertinência em cada uma das categorias.





3.1 Inferência Fuzzy

A inferência é o processo usado na avaliação das regras para a tomada de decisão. Nesse processo usa-se a lógica difusa (uma extensão da lógica convencional) na avaliação dos operadores usados nas regras. Os operadores lógicos E e OU são avaliadados atráves das chamadas Normas-T e Normas-S, as normas mais utilizadas usam os procedimentos min-max, basicamente estes procedimentos calculam o máximo e o mínimo entre a pertinência de duas variáveis, sendo min usando para a norma-T e max para norma-S.

Vejamos um exemplo de inferência aplicado no nosso sistemas para a tomada de decisão de ligar 40% das bombas:

Erro (%) / **Nível do tanque**	NMG	NG	NM	NP	Aceitável	PP	PM	PG	PMG
MA	Ligar todas as bombas da estação								
A	Desligar 50%*	Desligar 30%*	Desligar 20%*	Desligar 20%*	Manter a programação de bombas	Ligar 30% **	Ligar 40%**	Ligar 40% **	Ligar 50% **
M									
B		Desligar 40%*	Desligar 30%*	Desligar 20%*		Ligar 20% **	Ligar 30%**	Ligar 40% **	
		Desligar 40%*	Desligar 40%*	Desligar 30%*		Ligar 20% **	Ligar 20%**	Ligar 30% **	
MB	Desligar todas as bombas da estação								

Regra	***Enunciado***	***Pertinência Fuzzy***
1	Se Erro é PG e	u = 0,6
	nível é A	u = 0,1
	Entao	min[0,6 ; 0,1] = 0,1
2	Se Erro é PG e	u = 0,6
	nível é M	u = 0,2
	Entao	min[0,6 ; 0,2] = 0,2
3	Se Erro é PG e	u = 0,6
	nível B	u = 0,5
	Entao	min[0,6 ; 0,5] = 0,5

Por fim para descrobrir a pertinência para a categoria escolhida (ligar 40 % das bombas) é operada a norma-S e então o max= [0,1 ; 0,2 ; 0,5] = 0,5, ou seja tomaríamos a decisão de desligar 40% das bombas com um grau de pertinência de 50 %. Durante o procedimento de tomada de decisão as outras regras são avaliadas juntamente com o grau de pertinência de cada uma das regras. Com isso o sistema torna-se mais robusto, veja que para terminadas valores das variáveis de entrada que estejam perto do limiar de mudança de categoria será dado um valor de pertinência para ambas as categorias, o que no caso de sistemas não-fuzzy só poderíamos estar em uma ou outra categoria, o que poderia ser facilmente afetados por imprecisões na definição do modelo por exemplo, ou até mesmo por erros no processo que faz a aquisição de dados da planta indústrial que se quer controlar.

Conclusão

Um dos pontos principais deste trabalho foi construir uma implementação que pudesse facilitar o desenvolvimento da operação via regras principalmente atráves de um modelo que acelerasse o desenvolvimento num ambiente com mudanças constantes de requisitos, tipicamente o que ocorre numa equipe que desenvolve pesquisa. Este objetivo foi alcançado plenamente de acordo com a boa recepção da solução pela equipe de engenharia. Já estamos planejando desenvolver outros módulos do software usando a programação baseada em regras e com suporte do framework *Drool*, um desses módulos é um motor de inferência que analize a consistência dos dados do supervisório que monitora dados da planta de produção de pétroleo com a qual trabalhamos.

Um importantes trabalho futuro é a implementação e avaliação do sistema fuzzy proposto pelo autor em comparação com o sistema inicial, verificando aspectos de robustez e considerando outros objetivos além do padrão de vazão constante, como por exemplo o custo da energia gasto durante o processo de produção.

Referência Bibliográficas

GALVÃO, C.O; VALENÇA, M. J. A. *SISTEMAS INTELIGENTES Aplicações a Recursos Hídricos e Ciências Ambientais.* Porto Alegre; Ed. Universidade/UFRGS/ABRH, 1999.(Coleção ABRH de Recursos Hídricos; 7.)

SHAW, S. S; SIMÕES, M. G. *Controle e modelagem Fuzzy*. Editora Edgard Blüncher

NASCIMENTO Jr,. C. L.; YONEYAMA, T. *Inteligência Artificial em Controle e automação.* Editora Edgard Blüncher

MACHADO, E. **Operação de Redes de Escoamento de Petróleo Utilizando Algoritmo Genético Multi-Objetivo** Dissertação (Mestrado em Engenharia Civil e Ambiental) - Pós-Graduação em Engenharia Civil e Ambiental da Universidade Federal de Campina Grande, junho de 2006.

Apêndice

Código usando o framework *Drools*, implementa um conjunto inicial de regras não-fuzzy para o escalonamento de bombas hidráulicas:

```
import com.sample.DroolsTest.Message;
import com.sample.EstacaoReceptora;;
import com.sample.Information;
import com.sample.Tanque;

/**
*       Operação Via-Regras version 1.0
*       Thiago Emmanuel Pereira da Cunha Silva, thiago.manel@gmail.com
*/

//Divisão por zero! vazão desejada é igual a zero
rule "XPTO Rule"
        when
                inf : Information(error == Double.MAX_VALUE)
                t : Tanque()
                man : ActionManager()
        then
                String saida = null;
                saida = nivel(t.getNivel(), t.getAltura());

                if(saida.equals("POS_ALTO")){
                        man.atuaBombas(true, 1);//liga todas as bombas da estacao
                }else {
                        man.atuaBombas(false, 1);//desliga todas as bombas da estacao
                }
                System.out.println( "Vazão desejada é igual a zero" );//logar
end

rule "Equals Rule"
        when
                inf : Information(error == 0)
                t : Tanque()
                man : ActionManager()
        then {
                String saida = null;
                saida = nivel(t.getNivel(), t.getAltura());

                if(saida.equals("POS_ALTO")){
                        man.atuaBombas(true, 1);//liga todas as bombas da estacao
                }else if(saida.equals("NEG_ALTO")){
                        man.atuaBombas(false, 1);//desliga todas as bombas da estacao
                }else {
                        //do nothing! Mantém a programação
                }
                System.out.println( "Vazão desejada é igual a monitorada" );//logar
        }
end
```

```
rule "Positive Rule"
	when

		inf : Information(error > 0, error != Double.MAX_VALUE)
		t : Tanque()
		man : ActionManager()
	then
		String saida = null;
		saida = nivel(t.getNivel(), t.getAltura());

		if(inf.getError() > 0 && inf.getError() < 0.1){
			if(saida.equals("POS_ALTO")){
				man.atuaBombas(true, 1);//ligar todas as bombas
			}else if(saida.equals("POS_BAIXO")){
				//mantém a programação
			}else if(saida.equals("ZERO")){
				//mantém a programação
			}else if(saida.equals("NEG_BAIXO")){
				//mantém a programação
			}else if(saida.equals("NEG_ALTO")){
				man.atuaBombas(false, 1);//desligar todas as bombas
			}
		}
		else if(inf.getError() > 0.1 && inf.getError() < 0.3){
			if(saida.equals("POS_ALTO")){
				man.atuaBombas(true, 1);//ligar todas as bombas
			}else if(saida.equals("POS_BAIXO")){
				man.atuaBombas(true, 0.3);//liga 30% das bombas da estacao
			}else if(saida.equals("ZERO")){
				man.atuaBombas(true, 0.2);//liga 20% das bombas da estacao
			}else if(saida.equals("NEG_BAIXO")){
				man.atuaBombas(true, 0.2);//liga 20% das bombas da estacao
			}else if(saida.equals("NEG_ALTO")){
				man.atuaBombas(false, 1);//desligar todas as bombas
			}
		}
		else if(inf.getError() > 0.3 && inf.getError() < 0.6){
			if(saida.equals("POS_ALTO")){
				man.atuaBombas(true, 1);//ligar todas as bombas
			}else if(saida.equals("POS_BAIXO")){
				man.atuaBombas(true, 0.4);//liga 40% das bombas da estacao
			}else if(saida.equals("ZERO")){
				man.atuaBombas(true, 0.3);//liga 30% das bombas da estacao
			}else if(saida.equals("NEG_BAIXO")){
				man.atuaBombas(true, 0.2);//liga 20% das bombas da estacao
			}else if(saida.equals("NEG_ALTO")){
				man.atuaBombas(false, 1);//desligar todas as bombas
			}
		}
		else if(inf.getError() > 0.6 && inf.getError() < 1.0){
			if(saida.equals("POS_ALTO")){
				man.atuaBombas(true, 1);//ligar todas as bombas
			}else if(saida.equals("POS_BAIXO")){
				man.atuaBombas(true, 0.4);//liga 40% das bombas da estacao
			}else if(saida.equals("ZERO")){
				man.atuaBombas(true, 0.4);//liga 40% das bombas da estacao
			}else if(saida.equals("NEG_BAIXO")){
				man.atuaBombas(true, 0.3);//liga 30% das bombas da estacao
			}else if(saida.equals("NEG_ALTO")){
				man.atuaBombas(false, 1);//desligar todas as bombas
			}
		}
```

```
                else if(inf.getError() >= 1.0){
                        if(saida.equals("POS_ALTO")){
                                man.atuaBombas(true, 1);//ligar todas as bombas
                        }else if(saida.equals("POS_BAIXO")){
                                man.atuaBombas(true, 0.5);//liga 50% das bombas da estacao
                        }else if(saida.equals("ZERO")){
                                man.atuaBombas(true, 0.5);//liga 50% das bombas da estacao
                        }else if(saida.equals("NEG_BAIXO")){
                                man.atuaBombas(true, 0.5);//liga 50% das bombas da estacao
                        }else if(saida.equals("NEG_ALTO")){
                                man.atuaBombas(false, 1);//desligar todas as bombas
                        }
                }
                System.out.println( "Vazão monitorada é maior que a desejada" );
end


rule "Negative Rule"
        when

                inf : Information(error < 0)
                t : Tanque()
                man : ActionManager()
        then
                String saida = null;
                saida = nivel(t.getNivel(), t.getAltura());

                if(inf.getError() < 0 && inf.getError() > -0.1){
                        if(saida.equals("POS_ALTO")){
                                man.atuaBombas(true, 1);//ligar todas as bombas
                        }else if(saida.equals("POS_BAIXO")){
                                //mantém a programação
                        }else if(saida.equals("ZERO")){
                                //mantém a programação
                        }else if(saida.equals("NEG_BAIXO")){
                                //mantém a programação
                        }else if(saida.equals("NEG_ALTO")){
                                man.atuaBombas(false, 1);//desligar todas as bombas
                        }
                }
                else if(inf.getError() < -0.1 && inf.getError() > -0.3){
                        if(saida.equals("POS_ALTO")){
                                man.atuaBombas(true, 1);//ligar todas as bombas
                        }else if(saida.equals("POS_BAIXO")){
                                man.atuaBombas(false, 0.2);//desliga 20% das bombas da estacao
                        }else if(saida.equals("ZERO")){
                                man.atuaBombas(false, 0.2);//desliga 20% das bombas da estacao
                        }else if(saida.equals("NEG_BAIXO")){
                                man.atuaBombas(false, 0.3);//desliga 30% das bombas da estacao
                        }else if(saida.equals("NEG_ALTO")){
                                man.atuaBombas(false, 1);//desligar todas as bombas
                        }
                }
                else if(inf.getError() < -0.3 && inf.getError() > -0.6){
                        if(saida.equals("POS_ALTO")){
                                man.atuaBombas(true, 1);//ligar todas as bombas
                        }else if(saida.equals("POS_BAIXO")){
                                man.atuaBombas(false, 0.2);//desliga 20% das bombas da estacao
                        }else if(saida.equals("ZERO")){
                                man.atuaBombas(false, 0.3);//desliga 30% das bombas da estacao
                        }else if(saida.equals("NEG_BAIXO")){
                                man.atuaBombas(false, 0.4);//desliga 40% das bombas da estacao
                        }else if(saida.equals("NEG_ALTO")){
                                man.atuaBombas(false, 1);//desligar todas as bombas
                        }
```

```
                }




                else if(inf.getError() < -0.6 && inf.getError() > -1.0){
                        if(saida.equals("POS_ALTO")){
                                man.atuaBombas(true, 1);//ligar todas as bombas
                        }else if(saida.equals("POS_BAIXO")){
                                man.atuaBombas(false, 0.3);//desliga 30% das bombas da estacao
                        }else if(saida.equals("ZERO")){
                                man.atuaBombas(false, 0.4);//desliga 40% das bombas da estacao
                        }else if(saida.equals("NEG_BAIXO")){
                                man.atuaBombas(false, 0.4);//desliga 40% das bombas da estacao
                        }else if(saida.equals("NEG_ALTO")){
                                man.atuaBombas(false, 1);//desligar todas as bombas
                        }
                }
                else if(inf.getError() >= -1.0){
                        if(saida.equals("POS_ALTO")){
                                man.atuaBombas(true, 1);//ligar todas as bombas
                        }else if(saida.equals("POS_BAIXO")){
                                man.atuaBombas(false, 0.5);//desliga 50% das bombas da estacao
                        }else if(saida.equals("ZERO")){
                                man.atuaBombas(false, 0.5);//desliga 50% das bombas da estacao
                        }else if(saida.equals("NEG_BAIXO")){
                                man.atuaBombas(false, 0.5);//desliga 50% das bombas da estacao
                        }else if(saida.equals("NEG_ALTO")){
                                man.atuaBombas(false, 1);//desligar todas as bombas
                        }
                }
                System.out.println( "Vazão desejada é maior que a monitorada" );

end

/**
*       Funcao que define a relação entre nivel da coluna de fluido e
*       altura do tanque.
*/
function  String nivel(double nivel, altura){
        if(nivel < 0.6*altura && nivel >= 0.4*altura){
                return "ZERO";
        }
        else if(nivel < 0.4*altura && nivel >= 0.1*altura){
                return "NEG_BAIXO";
        }
        else if(nivel < 0.1*altura){
                return "NEG_ALTO";
        }
        else if(nivel < 0.9*altura && nivel >= 0.6*altura){
                return "POS_BAIXO";
        }
        return "POS_ALTO";
}
```