Epecificação formal e prototipação de controlador de sitema de bombeamento de água

Gabriela Moreira Mafra[1]

[1]Departamento de Ciência da Computação
Centro de Ciências Tecnológicas
Universidade do Estado de Santa Catarina (UDESC)
Joinville, SC – Brasil

{gabrielamoreiramafra@gmail.com}

Resumo. *Estações de bombeamento de água consomem energia em alto volume, abrindo espaço para otimização desse custo energético de acordo com as variações do uso da água. O controle das bombas com esse objetivo é feito sob diversas restrições, que precisam ser validadas por diferentes entidades. Propõe-se uma especificação de sistema de controle para tais estações em uma liguagem de especificação formal, fornecendo vantagens de verificação de restrições. Um protótipo executável gerado a partir da especificação e adiciona-se, por fim, um módulo de comunicação MQTT a fim de tornar o protótipo viável em uma rede de sensores de baixo consumo.*

Abstract. *Water pump stations have a high energy consumption volumes, opening room for enery cost optimization according to variations on water usage. Pumo control with this objective is done under many restrictions, which need to be validated by different enitities. A control system specification for this stations is proposed using a formal specification language, providing advantages on restriction validations. An executable prototype is generated from the specification, and lastly, a comunnication module using the MQTT protocol is added, making it viable for the prototype to be used whithin a low power sensor network.*

Keywords. *Internet of Things, Formal Methods, Pump Control, Temporal Logic of Actions, MQ Telemetry Transport*

1. Introdução

Conforme [Feng et al. 2018], o gasto com energia elétrica em estações de bombeamento representam o maior gasto de energia elétrica de algumas indústrias como no setor metalúrgico. O controle das bombas de água precisa manter os níveis de água dentro de uma margem estabelecida, ao passo que esse mesmo nível varia conforme o uso. O custo energético se dá pelo ato de desligar e ligar as bombas, assim como mantê-las ligadas. Visando minimizar tal custo, os mecanismos de controle avaliam as mudanças no nível de água e determinam quais bombas ligar ou desligar.

Os sistemas de bombeamento podem ser significantemente heterogêneos, como é o caso da estação estudada em [Borkowski, Wetula e Bień 2012]. Nessa estação, bombas diferentes tem custos associados diferentes, e há uma necessidade de equiílibrio de uso

entre bombas semelhantes. Os autores propõem um sistema de controle considerando essas restrições, porém não apresentam nenhuma evidência ou prova de que foram atendidas além da própria interpretação do código. O atendimento a essas restrições precisou ser avaliado por governos locais, tamanha é sua importância.

Nesse cenário, propõe-se uma forma alternativa de definição desse sistema de bombeamento, dada pela representação em uma linguagem de especificação formal. Nessa estrutura, é possível validar diversas propriedades, tais quais as restrições impostas, verificando que estas são atendidas em todos os estados possíveis dentro do especificado.

Utilizando a linguagem de especificação TLA$^+$ - que tem sido difundida e possui casos de aplicação relevantes como na Amazon [Newcombe et al. 2015] - é possível explicitar as restrições na própria especificação, usando uma sintaxe matemática familiar. A principal vantagem da utilização dessa linguagem para a representação é tornar mais fácil e confiável a validação do atendimento às restrições.

Uma especificação formal, por si só, não permite uma execução em um sistema real ou simulado para uma validação empírica ou para a própria aplicação real. Mas utilizando um tradutor como o proposto em [Mafra 2019], se faz possível a obtenção de um protótipo na linguagem de programação Elixir, que por sua vez pode ser executado e aplicado.

Assim, propõe-se a especificação formal para o sistema de controle apresentado em [Borkowski, Wetula e Bień 2012], a geração de código a partir dessaa especificação e a adaptação desse protótipo para um sistema real, onde os elementos se comunicam com o protocolo MQTT.

1.1. Objetivo

O objetivo deste trabalho se dá pela geração de um protótipo de sistema de controle de bombeamento de água equivalente ao proposto em [Borkowski, Wetula e Bień 2012] a partir de uma especificação formal do algoritmo de controle.

1.1.1. Objetivos Específicos

- Especificar o algoritmo de controle proposto em [Borkowski, Wetula e Bień 2012] usando a liguagem TLA$^+$;
- Gerar código executável em Elixir a partir dessa especificação, usando a ferramenta proposta em [Mafra 2019];
- Adicionar módulo de comunicação com protocolo MQTT para receber dados de sensores;
- Simular o envio de dados de sensores, verificando se as reações do controlador estão de acordo com o esperado para o algoritmo.

2. Trabalhos Relacionados

O principal trabalho relacionado a este é a própria proposta do algoritmo aqui especificado [Borkowski, Wetula e Bień 2012]. No trabalho, os diversos fatores de uma estação de bombeamento especifica, localizada na China, são considerados em uma otimização do processo de controle das bombas de água. Para obter o algoritmo, é feita uma simulação

do ambiente, onde o fluxo de água é simulado a partir da capacidade das bombas, e um algoritmo genético faz a seleção do controlador ótimo. Os principais elementos do algoritmo resultante é dado na notação ANSI C.

Outros trabalhos envolvendo Internet das coisas e controle de estações de bombeamento já foram feitos, indicando a relevância desse tipo de aplicação. [Feng et al. 2018] propõe um sistema de monitoramento de saúde e sustentabiliade de vários elementos da estação, envolvendo centenas de sensores. O estudo em [Choi 2013] propõe um sistema de controle para bombeamento de água quente, onde existem novas otimizações a serem feitas, como a mantenabilidade da temperatura da água.

Em termos de especificar formalmente esse tipo de sistema, um trabalho que tangibiliza a ideia é feito em [Attiogbé, Poizat e Salaün 2003]. Os autores propõe o uso de uma combinação entre métodos de especificação formais (utilizando Z e B) e semi-formais como UML ou SDL. Eles então aplicam essa ideia em um estudo de caso para uma estação de gás, que possui elementos comuns com estações de bombeamento de água, como as próprias bombas.

3. Abordagem Proposta

Dados os objetivos específicos, apresenta-se aqui o processo de elaboração e uma descrição detalhada de cada um dos artefatos produzidos.

3.1. Especificação Formal

O primeiro passo para a nova representação e prototipação do algoritmo de controle base (dado em [Borkowski, Wetula e Bień 2012]) é a própria especificação formal em TLA$^+$ desse algoritmo. A tradução é feita de forma manual e é complexa, já que se tratam de estruturas de representação bastante diferentes. A representação usada na proposta original, ANSI C, é procedural; enquanto a representação em TLA$^+$ é declarativa. Cada ação, dada por uma sequência de passos, é transformada em na definição de transições de estado válidas.

Algumas restrições tratadas na representação original de forma implícita são deixadas explícitas na especificação em TLA$^+$. Um exemplo é a prioridade das bombas, que é originalmente definida pela ordem de iteração $i = 0..n$ e, em TLA$^+$, é dada como uma das condições para a transição ser válida. A especificação dessa priorização em ANSI C se econtra na Figura 1, e a versão em TLA$^+$ se encontra na Figura 2.

Na especificação em TLA$^+$, se explicita que as bombas de número 4 acima só podem ser ativadas se as bombas com menor custo associado (de 1 a 3) estiverem desligadas. Isto é garantido pela condição $\forall i \in 0..2 : states[i] \notin \{"OFF"\}$.

Uma vez que a especificação foi finalizada, algumas verificações simples foram rodadas com o model checker TLC, verificando que o comportamento era conforme o esperado e fazendo os ajustes necessários. Destaca-se que esse processo de tradução manual é onde se concentra a maior parte do trabalho, enquanto o restante é obtido a partir dessa especificação.

3.2. Geração de Código

Com a especificação finalizada, é possível gerar código na linguagem de programação Elixir com a ferramenta proposta em [Mafra 2019]. O trabalho justifica a escolha da

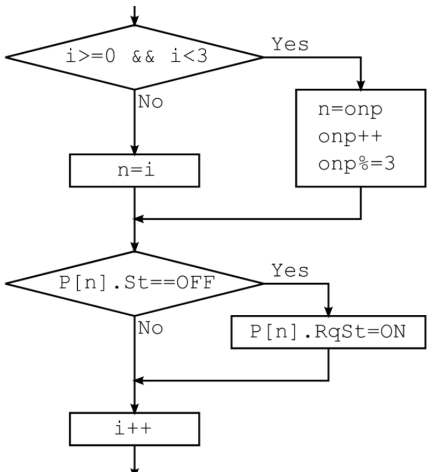

Figure 1. Especificação da priorização de bombas a serem ativadas em ANSI C. Fonte: [Borkowski, Wetula e Bień 2012]

$$
\begin{aligned}
activate(p) \triangleq\ & \land\ states[p] = \text{“OFF”} \\
& \land\ requestedStates' = [requestedStates \text{ EXCEPT } ![p] = \text{“ON”}] \\
& \land\ \text{UNCHANGED } \langle ofp \rangle \\
& \land\ \text{IF } (p \geq 0 \land p < 3) \\
& \quad \text{THEN } \land\ onp = p \\
& \qquad\ \ \land\ onp' = (p+1)\%3 \\
& \quad \text{ELSE } \land\ \forall\, i \in 0\,..\,2 : states[i] \notin \{\text{“OFF”}\} \\
& \qquad\ \ \land\ onp' = onp
\end{aligned}
$$

Figure 2. Especificação da priorização de bombas a serem ativadas em TLA⁺ . Fonte: autora

liguagem Elixir para a tradução com pontos relevantes, como a proximidade com a especificação, que torna o código mais fácil de entender e correlacionar com o artefato original; e a portabilidade dada pela máquina virtual de Erlang BEAM (usada por Elixir). Este segundo ponto é de extrema valia para o trabalho aqui proposto, já que possibilita que o protótipo gerado seja executado em dispositivos condizentes com o contexto, como um Raspberry Pi.

Para gerar o código, vários ajustes foram necessários na ferramenta de tradução. A ferramenta é apenas um protótipo inicial, e não suporta toda a gramática da linguagem TLA⁺ . Assim, novas regras de tradução precisaram ser criadas, contribuindo para a maturidade da ferramenta. Com todos os ajustes feitos, o protótipo pode ser gerado com sucesso, sem necessidade de alterações manuais no arquivo gerado.

3.3. Comunicação

O código gerado prevê a implementação de uma forma de comunicação com entradas externas, provido pela própria ferramenta de tradução. Essa comunicação é feita por um processo denominado oráculo, que é responsável por receber entradas sempre que o sistema em si não conseguir determinar o que fazer - isto é, quando se necessita uma entrada externa. No caso da estação de bombeamento, as entradas externas são:

- Aumento ou diminuição do nível de água do reservatório;

- Estado de cada uma das bombas de água (saudável ou danificado).

O oráculo é um processo a parte, que se comunica com o sistema através de canais do Elixir, que podem se comunicar com através da rede. Esse tipo de comunicação não seria adequado para receber informações de sensores simples, que podem não ser capazes de executar uma máquina virual BEAM. Considerando isso, é adicionado um novo módulo de comunicação que recebe as leituras dos sensores utilizando o protocolo MQTT. Esse módulo pode ser executado no próprio hardware do controlador, dentro de uma BEAM, sendo assim capaz de se comunicar com o oráculo pelos canais de Elixir.

A utilização do protocolo MQTT se justifica pela combinação de sua compatibilidade em sistemas de Internet das coisas e sua interface simples do tipo Publicador-Subscritor. O pacote Tortoise [Gausby 2018] oferece um cliente MQTT para Elixir, usado para inscrever o módulo de comunicação nos tópicos usados pelos sensores. Como broker, utilizou-se o Mosquitto [Light 2020], executado também no controlador.

Os sensores são simulados, uma vez que a obtenção de dados reais ou experimentação no ambiente não são viáveis. Assim, a publicação das mensagens de leitura dos sensores é enviada pelo próprio cliente do Mosquitto, com o executável `mosquitto_pub`.

Usando o padrão do protocolo MQTT, os tópicos foram criados como:

- `water_level` recebendo mensagens "up" ou "down"
- `pumps/+/state` recebendo mensagens "healthy" ou "damaged"

A leitura do nível de água é abstraída para facilitar a simulação. No ambiente real, haveriam leituras dos 11 sensores dispostos no reservatório, informando se a leitura é "com água" ou "sem água". A modificação para esse modelo seria trivial, e a motivação para a simplificação em uma única leitura de "up" ou "down" garante que as alterações simuladas sejam condizentes com a realidade. Assim, quando uma leitura "up" é feita, considera-se que o nível da água aumentou a distância entre os dois sensores na região de alteração, uma vez que essas seriam as mudanças de fato detectadas no sistem real.

4. Resultados

O sistema final inclui 3 processos Elixir e um broker MQTT a serem rodados no controlador, e a publicação de leituras de sensores por um cliente MQTT qualquer. Os testes foram simulados em um ambiente com rede Ethernet, devido a limitações de hardware disponíveis, mas podem ser utilizados com qualquer interface de rede. A comunicação entre o controlador e os sensores foi feita com TCP/IP, usado pelo protocolo MQTT. As ações são impressas na tela do próprio controlador para a simplificação do escopo, e é trivial usar o mesmo cliente, Tortoise, para comunicar as ações com o protocolo MQTT em um novo tópico aos atuadores que estariam disponíveis no ambiente real.

O controlador pode ser executado em qualquer sistema com suporte à máquina BEAM e a um broker MQTT. Um dispositivo recomendado é o Raspberry Pi, com seu sistema operacional próprio que atende essas restrições. Destaca-se que o tipo de hardware e local onde o controlador está sendo executado é transparente, uma vez que toda comunicação entre controlador e sensores é feita através do protocolo MQTT.

A especificação formal em TLA$^+$ e todo o código para o controlador, incluindo o módulo de comunicação, se encontra disponível em `https://github.com/GabrielaMafra/pump-station`.

5. Considerações Finais

A conversão de representação do algoritmo de controle para uma linguagem de especificação se mostrou benéfica por explicitar restrições do sistema e perimitir validações de diversas propriedades. Adicionalmente, a geração de código a partir dessa especificação economiza o trabalho manual que seria necessário para implementar uma versão executável do algoritmo de controle - necessária quando este é especificado em outras representações não formais.

O protótipo gerado, em conjunto com o módulo de comunicação MQTT, se mostraram funcionais nos testes realizados. Os testes, contudo, foram realizados com dados arbitrários, já que os dados reais de leituras de sensores na estação considerada não estão disponíveis.

Considera-se este trabalho como uma exemplificação de uma forma mais robusta de propor soluções, aplicando ao contexto de Internet das coisas. Os benefícios proporcionados por especificar formalmente os algoritmos são ainda mais notáveis em sistemas complexos e distribuídos, sendo este caso de aplicação uma demonstração do procedimento e das resultantes de usar especificação formal em conjunto da geração de código.

5.1. Trabalhos Futuros

Na vertente de aproveitar por completo a especificação formal do algoritmo de controle, seria interessante listar uma série de propriedades desejadas nesse tipo de sistema e verificá-las para o algoritmo proposto. Neste trabalho, só foram especificadas propriedades específicas aos estados de execução, inseridas na própria especificação. Verificações mais complexas podem ser feitas com o uso de fórmulas de lógica temporal.

Em termos de exploração e fomentação da prática de especificar software formalmente, seria relevante ver trabalhos análogos a este em diferentes áreas de aplicação, principalmente em sistemas distribuídos, onde o valor gerado pelas verificações é ainda maior.

References

ATTIOGBé, C.; POIZAT, P.; SALAüN, G. Specification of a gas station using a formalism integrating formal datatypes within state diagrams. In: . [S.l.: s.n.], 2003. p. 240.

BORKOWSKI, D.; WETULA, A.; BIEń, A. Design, optimization, and deployment of a waterworks pumping station control system. *ISA Transactions*, v. 51, n. 4, p. 539 – 549, 2012. ISSN 0019-0578. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0019057812000316>.

CHOI, J. M. Study on the lwt control schemes of a heat pump for hot water supply. *Renewable Energy*, v. 54, p. 20 – 25, 2013. ISSN 0960-1481. AFORE 2011(Asia-Pacific Forum of Renewable Energy 2011). Disponível em: <http://www.sciencedirect.com/science/article/pii/S0960148112006027>.

FENG, H. et al. Internet of thing system to extract hierarchical healthy and efficiency information for pump station optimization. In: *Proceedings of the 2018 2nd International Conference on Big Data and Internet of Things*. New York, NY, USA: Association for Computing Machinery, 2018. (BDIOT 2018), p. 162–166. ISBN 9781450365192. Disponível em: <https://doi.org/10.1145/3289430.3289466>.

GAUSBY, M. *Tortoise*. 2018. Disponível em: <https://github.com/gausby/tortoise>. Acesso em: 21 jul. 2020.

LIGHT, R. *Mosquitto man page*. 2020. Disponível em: <https://mosquitto.org/man/mosquitto-8.html>. Acesso em: 21 jul. 2020.

MAFRA, G. M. *Tradução automática de especificação formal modelada em TLA+ para linguagem de programação*. 69 p. Monografia (Trabalho de Conclusão de Curso) — Universidade do Estado de Santa Catarina, Joinville, 2019.

NEWCOMBE, C. et al. How amazon web services uses formal methods. *Commun. ACM*, ACM, New York, NY, USA, v. 58, n. 4, p. 66–73, mar. 2015. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/2699417>.