Desenvolvimento de arquitetura de software para modelagem de bioprocessos

Autores: Priscila Marques da Paz[1] ; Caroline Satye Martins Nakama[2] ; Galo Antonio Carrillo Le Roux[1*]

[1] *Departamento de Engenharia Química, Escola Politécnica, Universidade de São Paulo*

[2] *Departamento de Engenharia Química, Norwegian University of Science and Technology*

Autor Correspondente: [*]galoroux@usp.br

Resumo

No contexto da Engenharia de Sistemas em Processos (PSE), desenvolver ferramentas que integrem informações experimentais com modelagem matemática é fundamental para aumentar a produtividade de um processo biotecnológico. Dessa maneira, este trabalho tem como objetivo desenvolver uma arquitetura de software para a modelagem de bioprocessos que seja acessível a um grupo multidisciplinar. Para isso, ela deve ser cuidadosamente projetada com base em uma ontologia que descreva os bioprocessos. A representação ontológica é realizada por diagramas da Linguagem de Modelagem Unificada (UML), cujo uso é demonstrado por um estudo de caso de estimação de parâmetros. Conclui-se que a arquitetura pode servir como modelo ou exemplo de boas práticas de desenvolvimento de software para orientar simulações e estimativas de parâmetros de bioprocessos de forma estruturada.

Palavras-chave

Bioprocessos, Ontologia, Software, UML.

1. Introdução

Os microrganismos são o coração de um processo biotecnológico, pois são os agentes responsáveis pela síntese de compostos complexos de forma sustentável. Entretanto, para tornar a produção economicamente viável, é necessário projetar o bioprocesso e usar ferramentas de PSE (Doran, 2013; Hemmerich et al., 2021). Além disso, as técnicas de aquisição de dados experimentais têm evoluído ao longo dos anos, resultando no aumento também do tamanho dos conjuntos de dados coletados, os quais muitas vezes só podem ser processados e interpretados por meio do desenvolvimento de métodos quantitativos. Portanto, combinar as ferramentas de PSE com habilidades biotecnológicas torna-se fundamental, havendo a necessidade de desenvolver ferramentas que facilitem o trabalho e a colaboração em grupos multidisciplinares, o que concerne à área de biologia de sistemas (Meyer e Saez-Rodriguez, 2021).

A biologia de sistemas tornou-se essencial para lidar com o aumento da informação. À medida que novas informações são coletadas e técnicas desenvolvidas, é possível incorporar novos princípios estudados aos modelos, tornando-os cada vez mais completos e precisos (Bassalo e Gill, 2016; Meyer e Saez-Rodriguez, 2021). Esta é a razão pela qual a biologia de sistemas é considerada um tema multidisciplinar, indicando que raramente uma única pessoa terá um conhecimento profundo em todos os seus aspectos relevantes (Zuo e Zhao, 2018).

Nesse cenário, o desenvolvimento de estruturas computacionais torna-se desejável para orientar aplicações em modelagem de bioprocessos. Recomenda-se que a estrutura seja cuidadosamente projetada, pois pesquisadores com diferentes formações podem não apenas utilizar o software, mas também contribuir com novos modelos cada vez que houver atualizações nos estudos. Desta forma, a arquitetura de software torna-se tão importante quanto o seu desenvolvimento. Mesmo em casos simples, recomenda-se que todo o sistema seja estruturado antes de iniciar sua implementação, pois eles tendem a aumentar de tamanho, complexidade e escopo. Para garantir a (re)usabilidade do sistema, as ontologias são uma opção de apoio para organizar informações (Guedes, 2018). Ontologia é uma especificação de um conceito, e normalmente envolve classes, a forma que se relacionam e axiomas para descrever a semântica pretendida. É usada na ciência da computação e passou a ser difundida em áreas como medicina, biologia e química, destacando-se a ontologia OntoCAPE para a Engenharia Química (Marquardt et al., 2010). No entanto, ainda é escasso na área de bioprocessos. O trabalho de Duong-Trung e colaboradores (2023) sugere que uma ontologia apropriada pode impulsionar a área de bioprocessos no aprendizado de máquina, o que favorece a automação e modelagem de dados. Há ferramentas gratuitas que auxiliam na análise de dados de bioprocessos, como AnaBioPlus (Oliveira et al., 2017) e pyFOOMB (Hemmerich et al., 2021), mas que não fornecem uma ontologia completa do seu desenvolvimento, o que pode impedir que contribuições e melhorias sejam adequadamente realizadas.

O conhecimento representado pela ontologia permite instanciar componentes de bioprocessos, o que torna a informação prontamente compartilhada, uma vez que esses componentes são padronizados e organizados. Uma das ferramentas utilizadas para ontologias é a UML, amplamente reconhecida na indústria de software e que evita inconsistências de implementação (Aurora et al., 2020; Yurin e Dorodnykh, 2020). Pode se tornar complexo em projetos detalhados, mas é uma linguagem visual padronizada para facilitar a comunicação (Arcuri, 2018). Neste trabalho, métodos e técnicas que podem descrever bioprocessos são programados na linguagem Julia. O uso da UML para a generalização do software é demonstrado com um estudo de caso.

2. Metodologia

2.1. Estrutura e desenvolvimento da ontologia

A ontologia tem origem filosófica, em que há interesse na conceituação do mundo. Seu elemento básico é a classe, que representa uma coleção de componentes que compartilham características comuns. É organizada por hierarquia, na qual cada propriedade herdada é atribuída a subclasses. Componentes pertencentes à mesma classe são chamados de instâncias, e as características ou parâmetros da classe são os atributos. Cada atributo pode ser identificado por seu nome e possui um ou mais valores específicos para a classe a que pertence (Marquardt et al., 2010; Zhang et al., 2013).

Os diagramas UML, que representam a ontologia, foram implementados usando o software StarUML (versão 5.0.2). Dois tipos de diagramas foram escolhidos: de caso de uso e de classe. O primeiro apresenta uma visão externa da funcionalidade que o sistema deve oferecer aos usuários, sem detalhar como tal funcionalidade será implementada. Consiste em dois itens principais: atores e casos de uso. Os atores, representados por "bonecos", podem ser os usuários ou outros sistemas. As funcionalidades disponíveis aos atores são conhecidas como casos de uso, que são representados por elipses descrevendo a ação. As interações entre eles são representadas por linhas tracejadas, com ações opcionais ("extend") ou obrigatórias (contínuas e "include") (Guedes, 2018).

O segundo diagrama escolhido inclui classes, interfaces e relacionamentos. Cada classe é um conjunto de objetos, fornecendo especificação de atributos e operações que uma instância da classe pode concluir. Essas operações são tratadas como funções ou procedimentos, ou seja, como as ações vão acontecer. Os relacionamentos possuem um conceito semelhante ao do diagrama de caso de uso, porém mais variado e detalhado (Guedes, 2018).

2.2. Modelagem do bioprocesso

A abstração e a classificação dos componentes de bioprocesso são baseadas em equações da forma mais genérica possível, o que permite adições de conceitos mais complexos relacionados ao componente, seja na própria classe ou por meio de interações com outras criadas. Foram criadas classes principais para o modo de operação de um biorreator (*OperationMode*), crescimento (*CellGrowth*) e morte celular (*CellDeath*), aeração (*Aeration*), rendimento (*Yield*), equações diferenciais ordinárias discretizadas (*ode*) e estimadores para o caso de estimação de parâmetros (*estimators*). Para cada caso, classes-filhas também são criadas e herdam características da respectiva classe oriunda. Por exemplo, a abstração da classe OperationMode se baseia no balanço de massa geral, dado pela Equação 1 (Doran, 2013):

$$\frac{dM}{dt} = M_i - M_o + R_G - R_C \tag{1}$$

em que a derivada dM/dt é a massa acumulada dentro do sistema, sendo zero se o sistema estiver em estado estacionário. M_i é a vazão mássica de um componente que entra no reator, M_O é a que sai, R_G é a taxa mássica de geração de um componente, e R_C é a de consumo. As classes-filhas são criadas para componentes como biomassa, substrato e produto, variando em: modo contínuo (nomeada como *continuous* para biomassa), que considera toda a equação, podendo ter reciclo (*continuouswr*); batelada (*batch*), que tem M_i e M_O iguais a zero; e batelada alimentada (*fedbatch*), que M_O é zero.

Quando um bioprocesso é realizado por microrganismos, o R_G é dado por:

$$R_G = \mu_X X \tag{2}$$

em que μ_X é a taxa de crescimento específica. Uma situação semelhante pode ser encontrada para R_C com uma constante de morte específica.

Percebe-se que μ_X de R_G interage com a classe *CellGrowth* e, uma vez que μ_X pode ocorrer de diversas maneiras, como por exemplo, com inibição ou não de substrato ou produto, *CellGrowth* será composta por subclasses que representem todas essas alternativas para obter μ_X. Se o substrato S é limitante e as mudanças de concentrações de outros componentes não têm efeito em μ_X, equações como de Monod, Tessier, Contois ou outras podem descrever esse comportamento. Por outro lado, quando as concentrações de substrato ou produto são altas e substâncias inibidoras estão presentes no meio, μ_X depende da concentração do inibidor. Então, essas situações podem ser descritas por outras equações (Shuler et al., 2002; Alterthum et al., 2020).

A cinética de crescimento possui parâmetros estequiometricamente relacionados, como manutenção celular e coeficientes de rendimento, e também é influenciada pelas condições ambientais. Há um conjunto de fatores a serem considerados e consequentemente, as classes propostas interagem entre si, além de surgir novas subclasses dentre elas. Por exemplo, em processos aeróbicos, o oxigênio se torna um fator limitante para o crescimento e, portanto, a classe *Aeration* deve ser considerada, com subclasses que descrevam a transferência de oxigênio (OTR) da fase gasosa para a fase líquida e a taxa de consumo de oxigênio (OUR) (Shuler et al., 2002). Em suma, a arquitetura fornece diversas situações e caminhos para a montagem de modelos.

2.3. Estudo de caso

Os diagramas UML que descrevem o sistema foram utilizados para guiar a programação do software. A linguagem escolhida para a implementação do código foi a linguagem Julia, devido principalmente à sua elevada performance e flexibilidade dinâmica, além de ser de código aberto.

Para avaliar o funcionamento do código, a estimação de parâmetros de um bioprocesso é realizada. O estudo de caso em questão é descrito por Mostoufi e Constantinides (2023) e tem como foco a produção de penicilina. O cultivo é realizado pelo microrganismo *Penicillium chrysogenum* em um fermentador em batelada sob condições cuidadosamente controladas. A taxa de crescimento celular pode ser modelada pela lei logística, já implementada como uma das opções para se obter μ_X. Nesse caso, os parâmetros são chamados de k_1 e k_2. A concentração de biomassa é representada por X e R_g é o mesmo da Equação 2. Como não há menção à morte celular, assume-se que R_c é igual a zero.

Para a produção de penicilina, a expressão é similar à Equação 3, também anteriormente considerada no sofware e generalizada pela Equação 4, que descreve o balanço do produto P e é baseada na Equação 1. Entretanto, q_p e k_{DP} foram representadas por k_3 e k_4 no exemplo.

$$\frac{dP}{dt} = q_p X - k_{DP} P \tag{3}$$

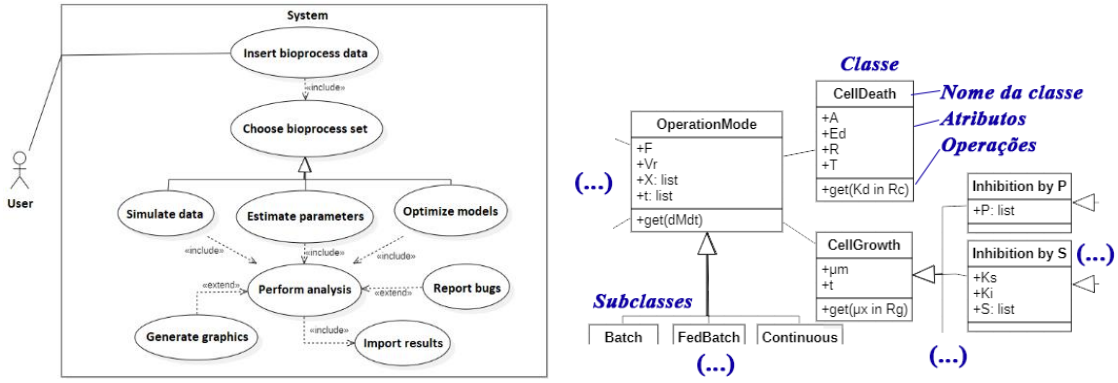
$$\frac{dP}{dt} = R_{G,P} - R_{C,P} \tag{4}$$

A média dos dados experimentais de X e P foi utilizada. O solver escolhido foi o IPOPT e a função objetivo consiste na minimização da soma dos erros ao quadrado entre os dados experimentais e os propostos pelo pelo modelo. As condições iniciais são de X = 0,29 %DW e P = 0 u/mL. As estimativas iniciais para os parâmetros k_1, k_2, k_3 e k_4 foram o h^{-1}.

3. Resultados e Discussão

3.1. Diagramas de UML

A arquitetura do software é mostrada pelos diagramas de classes (Figura 1-a) e o diagrama de caso de uso (Figura 1-b).



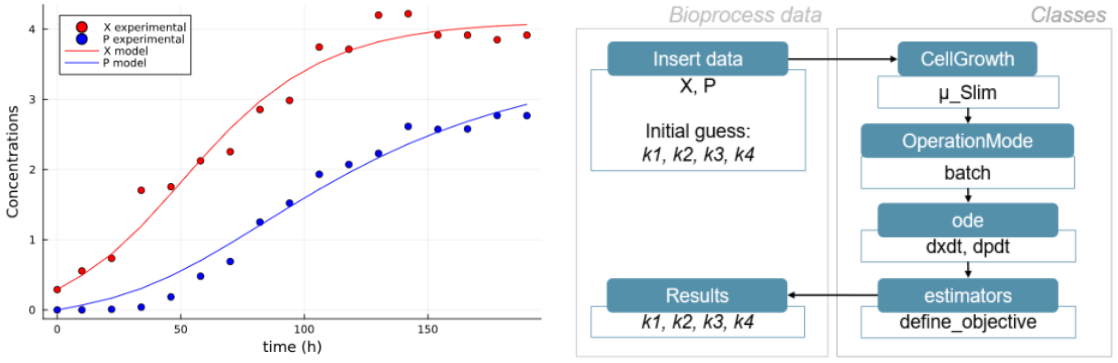
(a) (b)

Figura 1: UML para bioprocessos: (a) diagrama de casos de uso e (b) de classes.

A Figura 1-a mostra a cascata de ações, desde a inserção dos dados do bioprocesso no sistema até a geração e importação dos resultados. A ideia é que o usuário indique o que gostaria de fazer no sistema de acordo com os dados que possui: simular o bioprocesso, estimar parâmetros, ou mesmo otimizá-los. Já na Figura 1-b, as classes mencionadas anteriormente são representadas em um fragmento do diagrama total e os três pontos simbolizam que há mais subclasses consideradas. Isso indica que o diagrama de classes é dinâmico, assim como a arquitetura do software, e cresce dependendo do que for adicionado ao modelo.

3.2. Estimação de parâmetros da produção de penicilina

A Figura 2a mostra os valores experimentais e os obtidos pelo modelo de X e P. Os valores finais dos parâmetros estimados são $k_1 = 3,99\ h^{-1}$, $k_2 = 4,11\ h^{-1}$, $k_3 = 1,39\ h^{-1}$ e $k_4 = 1,61\ h^{-1}$. Um fluxograma com uma visão geral dos dados e funções que são chamados para estimativa de parâmetros é mostrado na Figura 2b.



(a) (b)

Figura 2: Estudo de caso: (a) concentrações de biomassa e penicilina e (b) fluxograma da estimação de parâmetros cinéticos.

O estudo de caso, para este trabalho, visa validar o funcionamento do software, principalmente através da (re)utilização de equações, ao invés de validar o modelo biológico em si. Além disso, neste exemplo específico, o foco de discussão dos autores foi o ajuste de curvas, não o significado biológico esperado pelos parâmetros. Contudo, vale destacar que a programação tal como foi feita, em conjunto à ontologia, permite que novas metodologias alternativas sejam incluídas e testadas, sendo esta a maior vantagem

em relação a um código gerado de maneira não estruturada. Cada classe pode chamar uma metodologia diferente (Figura 2b), mas que pertence a mesma classe, sem comprometer o código. No entanto, o aumento do número de equações no modelo pode tornar a diferenciação mais complexa. Por isso, o código em Julia foi ajustado para a biblioteca JuMP, permitindo maior flexibilidade e eficiência no processo de construção de modelos matemáticos. JuMP é uma linguagem de modelagem algébrica específica e possui uma sintaxe que imita expressões matemáticas naturais (Dunning et al, 2017). Por conter apenas equações algébricas, as equações diferenciais (EDO) foram discretizadas usando o algoritmo de Euler implícito, que é um método de colocação ortogonal de primeira ordem (Biegler, 2010). O seu uso impacta positivamente o desenvolvimento do software para lidar com problemas mais complexos.

4. Conclusão

A arquitetura desenvolvida do software pode trazer novas perspectivas para a modelagem de bioprocessos, já que atualmente a comunidade de software tem dado grande atenção a essa abordagem como uma abstração chave no processo de design. O uso da UML para generalização de software é demonstrado com um estudo de caso, mostrando que o software pode servir como modelo ou exemplo de boa prática de desenvolvimento para orientar simulações e estimativas de parâmetros de bioprocessos de forma estruturada. Em resumo, este protótipo é uma ferramenta alternativa para a integração e análise de dados baseada em modelos, que pode ser a semente de uma nova, robusta e sofisticada ferramenta para a biologia de sistemas.

Agradecimentos

Esta pesquisa foi apoiada financeiramente pela Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) nº. 88887.464619/2019-00, programa PROEX e Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pela bolsa Priscila M. da Paz (142147/2019-2) e a Bolsa de Produtividade 311550/2022-3.

Referências

A. Arcuri: An experience report on applying software testing academic results in industry: we need usable automated test generation, Empir. Software Eng. (23), 1959-1981, 2018.

F. Alterthum, W. Schmidell, U. A. Lima e I. O. Moraes: Biotecnologia Industrial - Fundamentos, volume I (2a Ed), Blücher: 2020.

V. Aurora, M. Singh e R. Bhatia: Orientation-based ant colony algorithm for synthesizing the test scenarios in uml activity diagram, Info. Soft. Tech. (123), 1-21, 2020.

M. Bassalo e R Gill: Directed evolution and synthetic biology applications to microbial Systems, Curr. Opin. Biotechnol. (39), 126-133, 2016.

L.T. Biegler: Nonlinear programming: concepts, algorithms, and applications to chemical processes, MOS-SIAM Series on Optimization: 2010.

P. Doran: Bioprocess engineering principles, Elsevier: 2013.

I. Dunning e J. Huchette, M. Lubin: JuMP: A modeling language for mathematical optimization, SIAM Review (59), 295-320, 2017.

N. Duong-Trung et al: When bioprocess engineering meets machine learning: A survey from the perspective of automated bioprocess development, Biochem. Eng. J. (190), 1- 21, 2023.

G. Guedes: UML 2 - Uma abordagem prática, Editora Novatec: 2018.

J. Hemmerich, N. Tenhaef, W. Wiechert e S. Noack: pyfoomb: Python framework for object-oriented modeling of bioprocesses, Eng. Life Sci. (21) 242-257, 2021.

W. Marquardt: OntoCAPE, Editora Springer Berlin Heidelberg: 2010.

P. Meyer e J. Saez-Rodriguez: Advances in systems biology modeling: 10 years of crowdsourcing dream challenges, Cell Syst. (12), 636–653, 2021.

N. Mostoufi e A. Constantinides: Chapter 8 - Linear and nonlinear regression analysis. In: Applied Numerical Methods for Chemical Engineers (N. Mostoufi e A. Constantinides Eds.), Academic Press: 2023.

C.M. Oliveira et al: AnaBioPlus: a new package for parameter estimation and simulation of bioprocesses, Brazilian J. Chem. Eng. (34), 1065-1082, 2017.

M. Shuler e F. Kargi: Bioprocess Engineering: Basic Concepts. Editora Pearson, 2002.

A. Yurin e N. Dorodnykh: Personal knowledge base designer: Software for expert systems prototyping, SoftwareX (11), 1–6, 2020.

J. Zhang e Hunter A, Zhou Y: A logic-reasoning based system to harness bioprocess experimental data and knowledge for design, Biochem. Eng. J. (74), 127–135, 2013.

Z. Zuo e K. Zhao: The more multidisciplinary the better? - the prevalence and interdisciplinarity of research collaborations in multidisciplinary institutions, J. Informetrics (12), 736–756, 2018.