CASO PRÁTICO DE PBL PARA ENSINO DA ENGENHARIA DE SOFTWARE: CONTROLE DE QUALIDADE ADICIONANDO VALOR POR CODIFICAÇÃO - DOCUMENTATION AND QUALITY ASSURANCE AS CODE

DOI: 10.37702/2175-957X.COBENGE.2025.6199

Autores: *REGINALDO ARAKAKI,OVIDIO LOPES,RENATO PENHA,KAIANE CORDEIRO,PAULA PIVA,ANNA ARAGÃO,BRUNA BRASIL ALEXANDRE*

Resumo: *This work describes how students encounter challenges in ensuring the quality of software systems by implementing quality control procedures. At first, students and some professionals think these activities are boring: documenting and applying test cases. No one likes this kind of activity. Professors and coaches created one dimension of challenges for students: code quality, code documentation, code control, and code rules. From a business view and technology viewpoint, everything is code. As a result, the system code created for quality control and quality assurance enabled students and business partners to develop a significant project related to the Quality Control PBL Module.*

Palavras-chave: *Architecture as code,Documentation as code,Functional and Non Functional requirements,PBL Project Based Learning,Software Engineering,Quality Assurance,Quality Control,Business Drivers,ATAM,ArchitectureTradeoff.*

CASO PRÁTICO DE PBL PARA ENSINO DA ENGENHARIA DE SOFTWARE: CONTROLE DE QUALIDADE ADICIONANDO VALOR POR CODIFICAÇÃO - DOCUMENTATION AND QUALITY ASSURANCE *AS CODE*

1 INTRODUÇÃO

A dinâmica do mercado e da indústria tem nos sistemas digitais, o canal fundamental para a disponibilização de produtos e serviços em todos os setores B2B (*Business-to-Business*), B2Gov (*Business-to*-Poder Público) e B2C (*Business-to-Consumer*). Há algum tempo, a necessidade de lançamento de novas funcionalidades está chegando ao lançamento de diversas versões por semanas ou por dia, ao invés de meses, nos ciclos mais longos de engenharia (Arakaki *et al*, 2020). Esta dinâmica de lançamento de sistemas de software exigem cuidados arquiteturais, para que as modificações não somente de código de software, como também de evoluções de infraestruturas de processamentos computacionais com muita frequência, sejam mantidas robustas. Se fosse válida uma comparação, seria a evolução de um corpo humano submetidas a intervenções cirúrgicas para correção ou para tratamento de contusões localizadas, porém, com perdas de capacidades de outros órgãos afetados. Em sistemas acontece algo muito similar. Ou seja, as intervenções de mudanças sobre partes do sistema trazem impactos que devem ser balanceados e aferidos em termos de custo/benefício: o aprimoramento de uma parte pode levar ao enfraquecimento de outras. Por exemplo, o *tradeoff*: Desempenho e Volume. Ao ampliar a capacidade de tratar maior volume de clientes, emerge o risco de lentidão.

Os aspectos de qualidade da arquitetura como a disponibilidade envolvem os chamados *Business Drivers* do sistema. Por exemplo, um aplicativo de carona, se ficar indisponível, perde transações de carona aos clientes, com impactos de negócio. Imagine-se um sistema de atendimento a um fluxo de internação hospitalar dentro de um pronto socorro lidando com casos emergenciais que possam trazer impactos à saúde de pessoas. Para isso, faz parte de controle arquitetural, aferir se os cenários como os indicados afetam a resiliência do sistema: a disponibilidade do sistema está sendo monitorada e, em caso de incidentes, dispõe de ações de alertas, alarmes e ações de recuperação por acionamento de mecanismos automáticos. Todos esses eventos estão com rastreabilidade? Ou seja, incluir o conhecimento e a documentação na forma de código de programas que podem ser executados, para aferir a qualidade.

Como transformar esse conhecimento em código? É muito comum, diante de crises de disponibilidade de sistemas, a restauração dos níveis de serviço depender muito de pessoas específicas que detém o conhecimento pela experiência, quase ignorando a documentação.

Tais conhecimentos incorporados no código se tornam ativos de software, pois têm a chance de serem mantidos como parte do sistema, validando a qualidade comportamental das funcionalidades e dos demais pilares que sustentam o sistema.

1.1 Objetivo do Artigo

O artigo documenta o caminho de aprendizagem em engenharia de software dos estudantes do Inteli, até chegarem neste módulo de Controle de Qualidade Arquitetural sob a ótica do *Project Based Learning* (PBL). Para isso, um caso prático onde os estudantes são submetidos ao projeto-desafio da automação do controle de qualidade do sistema pela arquitetura. Adicionalmente, ilustra-se casos de estudantes que chegam até esta fase do curso, pensando e agindo sobre qualidade somente para cumprir metas de notas: testes são maçantes, poucos úteis e provocam grande desperdício de tempo - muito similar a atitude de profissionais pressionados a entregas com baixa qualidade.

2 FUNDAMENTOS CONCEITUAIS

O estudante, visando o perfil do Engenheiro de Software, inicia os fundamentos de pequenos programas e integração com elementos de hardware e software, incluindo dispositivos para Internet das coisas (IoT), no seu primeiro ano de formação. Em seguida, no 2o ano e 3o ano, os módulos de aprendizagem ampliam o foco de codificação de pequenos programas e componentes para integração de sistemas. São arquiteturas com infraestruturas computacionais locais (*on premise*) conectados com plataformas de *cloud computing*. Acessando serviços por aplicativos em smartphones, notebooks, sensores e atuadores acionados por páginas, links, botões e até comandos por linguagem natural. São desafiados por arquiteturas com implementações robustas em termos de desempenho, escalabilidade, rastreabilidade, segurança, precisão e tolerâncias a falhas.

O perfil profissional do egresso do Curso de Engenharia de Software do Inteli está de acordo com as Diretrizes Curriculares Nacionais (DCNs) estabelecidas pela Resolução CNE/CES 5/2016, que institui as Diretrizes Curriculares Nacionais para os cursos de graduação na área de Computação. Além disso, as competências desenvolvidas no curso, expressadas no projeto pedagógico do curso (aprovado pelo MEC com nota 5), são ampliadas para atender às novas demandas do mercado de trabalho e da sociedade, proporcionando uma formação que articula as necessidades locais e regionais com as exigências globais, e o perfil profissional é ainda ampliado em função de novas demandas apresentadas pelo mundo do trabalho, como ilustrado no item 2.1 sobre as trajetórias de ensino, projetos e aprendizados aos estudantes.

Em cada um dos módulos PBL de Engenharia de Software, existe um metaprojeto direcionador que catalisa um projeto junto a um parceiro da indústria ou poder público, e os conceitos alinhados com as diretrizes estabelecidas no plano pedagógico. Obviamente, os conceitos são conduzidos por todo o conjunto de disciplinas de exatas, liderança e de negócios alinhadas com os fundamentos de engenharia de sistemas de modo a trazer a suficiência lógica para o desenvolvimento do projeto do módulo.

Tabela 2.1 - Meta projetos e aprendizado em arquitetura de sistemas (Fonte: autores).

Módulo	Meta Projeto	Abrangência Engenharia de Software
M5	Sistemas em Cloud Computing.	Processamento distribuído (incluindo infrastructure as code).
M6	Integração por Serviços (SOA).	Arquitetura de integração: visão componentizada de serviços e integração externa e legados.
M7	Interação e usabilidade PLN.	Usabilidade em computação pervasiva e interação assíncrona por linguagem natural.
M8	Arquitetura Tolerante a Falhas.	Mecanismos e táticas arquiteturais para robustez, segurança e resiliência de sistemas.
M9	Controle de Qualidade “*as Code*”.	Programas e sistemas que aferem a qualidade de sistemas, onde a codificação é o conhecimento.
M10	Fluxos de Integração Contínua.	Publicações de novas versões 2 vezes por ano, para 2 vezes por dia. Fluxos de desenvolvimento com controle de qualidade automatizados e ágil.

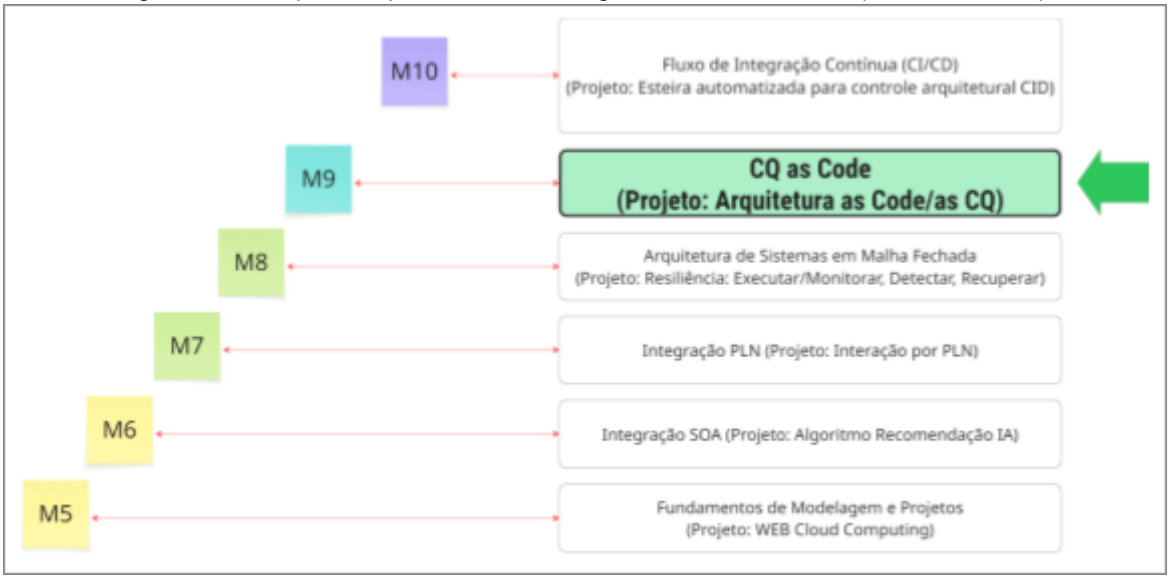
Fonte: elaborado pelos autores

As referências científicas que servem de pilares e arcabouços de conhecimentos e conceitos em todos esses módulos envolvem aspectos de qualidade de produto de software (Pressman, 2021) e (ISO/IEC 25010), onde os aspectos críticos mapeados em requisitos não funcionais podem ser aferidos estática e dinamicamente. Sobre o fluxo de desenvolvimento, abrangendo o ciclo de vida do sistema, alcançando os usuários finais, incidentes de ambiente de execução em produção (*runtime*) com nomenclatura indicada no mercado como DEVOPS e DEVSECOPS têm-se a referência de normas internacionais (ISO/IEC 12207). Ainda sobre a arquitetura de sistemas, focando produtos e processos, o detalhamento de um sistema por visões (ISO/IEC 10746; RM-ODP) que envolvem negócios, requisitos, mecanismos de engenharia e ferramentas de tecnologia organiza a forma como as decisões de arquitetura foram estabelecidas facilitando as avaliações (Táticas Arquiteturais SEI/CMU) e aferições de qualidade (ATAM/SEI/CMU). As táticas de controle de arquitetura considerando os conceitos de malha fechada de controle (Cruz, 2004) como referências em malhas fechadas de controle, mostrando aos alunos como muito das vulnerabilidades advém de mecanismos de controle em malha aberta ("funciona bem enquanto funciona… Quando dá pau, trava tudo e daí o time de operação entra com procedimentos do tipo stop/start e outros, com retentativas até restabelecer…"). Dessa forma, a implementação de uma malha fechada de controle para os requisitos do sistema – baseada na observação contínua, detecção proativa de desvios e ações corretivas automatizadas sobre os elementos da solução – é fundamental. Este mecanismo garante a resiliência do sistema com a máxima automação possível, eliminando a necessidade de intervenções manuais reativas e permitindo que o próprio sistema restabeleça rapidamente os níveis de serviço esperados.

2.1 Mapa de Aprendizagem da Engenharia de Software Inteli com PBL

Após o primeiro ano, também concebido com técnicas de ensino PBL, com projetos inspirados por desafios trazido por parceiros públicos ou privados do mercado, marcam o início da formação do perfil de engenheiro de software com foco no desenvolvimento de lógica computacional, fundamentos de integração de hardware e software, linguagens de programação e sistemas operacionais (Site Inteli, 2025), (Arakaki *et al*, 2021), (Barrows, 1996), (Cugnasca *et al*, 2024). Ver a figura 2.1.

Figura 2.1 - Mapa de Aprendizado de Engenharia de Software (Fonte: autores).



Fonte: elaborado pelos autores

No terceiro ano, onde está o módulo foco deste artigo, tem-se uma trajetória para o aprendizado da Engenharia de Software, como pode ser visto no desenho da figura 2.1 e tabela 2.1. São os módulos M5 até M11. Os meta-projetos PBL e os pilares de maneira sintetizada (Inteli PPC, 2024):

- **M5** - O projeto envolve um sistema web com desafios de conceitos e práticas como a criação de um aplicativo responsivo em estações internet e dispositivos móveis, com claros destaques em computação distribuída em nuvem (Cloud Computing) com camadas de interação humano-computador, lógica de negócios, base de dados e integração com legados internos e externos;
- **M6** - Neste módulo, a estrutura arquitetural do sistema enfatiza as integrações internas e externas, onde conceitos de serviços, conexão sem estados (stateless) das interfaces como as APIs, a rastreabilidade, disponibilidade, acessos simultâneos, integração assíncrona mostra aos estudantes os aspectos de escalabilidade dos sistemas dependem de decisões arquiteturais que atendam aos requisitos não funcionais. Destacam-se os conceitos de Service Oriented Architecture (SOA) e uso de algoritmos de recomendação usando técnicas de IA;
- **M7** - O meta-projeto deste módulo desafia os estudantes à adoção de interação humano-computador por linguagem natural tanto por textos, como por áudio e vídeo. Com isso, a integração com serviços em cloud computing de processamento de linguagem natural (PLN) são incorporados nos sistemas desenvolvidos, focando os conceitos anteriores mais a interação assíncrona e ferramentas de IA (Hayashi *et al*, 2020) e (Fujii *et al*, 2020). Os aspectos arquiteturais como disponibilidade, rastreabilidade, observabilidade e tolerância à falhas são aprimorados;
- **M8** - Evoluindo mais sobre aspectos não funcionais de um sistema, os alunos aprendem a diferenciar conceitos de arquitetura com base em fundamentos das normas ISO/IEC 10746, ISO/IEC 42010, ISO/IEC 25010, ATAM/SEI, (NIST, 2011) e táticas arquiteturais. Malha fechada de controle é base da resiliência de sistemas em disponibilidade, segurança, tolerância a falhas, rastreabilidade e outros. O estudante desenvolve visões críticas para entender que a vulnerabilidade do sistema não se resolve ampliando recursos de infraestrutura somente. Táticas de arquiteturas devem implementar mecanismos para “resistir”, “detectar” e “atuar” e garantir os níveis de serviços para os quais foram concebidos (Hayashi *et al*, 2020), (Hayashi *et al*, 2021), (Hayashi *et al*, 2023);
- **M9** - Módulo **foco** deste artigo. Como garantir que os aspectos não funcionais de um sistema estão bem construídos? As decisões arquiteturais conferem robustez aos requisitos do sistema? Gerar códigos que reflitam esses requisitos tanto em funcionalidade como em não funcionalidade foi o foco e tema deste artigo. Aos estudantes, que até esse momento do curso sempre vivenciou e praticou a documentação de sistemas na forma de textos em editores de documentos ou *markdown*, foram desafiados em transformar em código (*as code*) de controle de qualidade de aspectos como: regras de negócio, rastreabilidade, disponibilidade, acessos simultâneos, funcionalidades, segurança, tolerância a falhas, alertas preventivos e observabilidade;
- **M10** - Esse módulo tem como foco a esteira integrada DEVOPS, DEVSECOPS de automação, onde os elementos de controle de qualidade indicadas e desenvolvidas no módulo M9 podem ser inseridas nesta esteira. O destaque deste módulo é a escala da esteira. Alguns setores industriais hoje geram versões de software em produção por dia, diferentemente de 2 versões por ano de esteiras mais antigas, baseadas em processos manuais.

2.2 Módulo sobre Qualidade de Sistemas e Arquitetura

Conforme destacado no item 1.1, módulo M9, o aprendizado conceitual e profissional foca no controle de qualidade baseada em evidências estáticas e dinâmicas, direcionadas e priorizadas por visões de arquitetura (ISO/IEC 42010, ISO/IEC 25010, ISO/IEC 10746). Fazem parte da avaliação estática de código os conceitos de gestão de configuração, mapa de calor de mudanças, cobertura de testes entre outros. Do ponto de vista de avaliação dinâmica, o comportamento do sistema diante de cenários de normalidade e de cargas não funcionais como volumes, acessos simultâneos, disponibilidade. Incluem casos de testes simulando situações críticas para aferição do comportamento citado.

2.3 Projeto PBL - instanciação do meta projeto e execução

O módulo é dividido sempre em 5 sprints de entregas, por 10 semanas, a cada duas semanas, divididos da seguinte forma, com controle de qualidade *as code*:

- Sprint Review SPR1: Foco do projeto e descrição de regras de negócio (business drivers) como código, automatizando os fluxos críticos de negócio, com as devidas rastreabilidade e alimentação de painéis de qualidade (*quality dashboards*);
- Sprint Review SPR2: Foco nos requisitos funcionais e não funcionais como código, automatizando os fluxos críticos de arquitetura, com as devidas rastreabilidade deste controle de qualidade e alimentação de painéis de qualidade (*quality dashboards*);
- Sprint Review SPR3: Solução técnica avaliada em termos de códigos para verificar aspectos de integração com serviços de base de dados, serviços externos e demais serviços internos, incluindo os níveis de serviços contratados de infraestrutura computacional. Profissionalmente, a avaliação estática da solução ficaria mais completa, em termos de adoção de práticas de avaliação das táticas arquiteturais para inspirar a codificação para realizar as aferição dinâmicas com cenários que estimulem os mecanismos de código das táticas;
- Sprint Review SPR4: Consolidação do painel de controle de qualidade, alinhando as automações e resultados que subsidiam a aferição de qualidade baseadas nas dimensões de business, requisitos e solução técnica;
- Sprint Review SPR5: Revisão e ajustes dos itens construídos ao longo das Sprints anteriores, incluindo a geração de uma storyboard baseada no tema do módulo.

Os próximos itens do artigo evidenciam o projeto realizado junto a um parceiro de operação logística de entregas de alimentos, produtos e outras conveniências para clientes finais (B2C) e clientes empresas (B2B).

3 PBL - SISTEMA LOGÍSTICO

O rito PBL (*Project Based Learning*) foco deste artigo corresponde ao módulo M9 (ver figura 2.1), onde um parceiro de mercado atuando na operação logística de ecommerce, promove condições de conveniências para clientes pessoas físicas e clientes pessoas jurídicas, focando em entregas na “última milha” (jargão do fluxo logístico que inclui desde a coleta, distribuição e entrega na residência do cliente. Geralmente, nesta última etapa, realizada por pacotes conduzidos por motoqueiros engajados no conceito de colaboradores autônomos alocados por um processo muitas vezes chamado de “uberização”. O aplicativo de apoio aos entregadores é o foco do projeto PBL, desafiando os alunos ao entendimento, aferição da arquitetura e proposição do controle de qualidade do sistema.

3.1 Domínio do Problema - Escopo do Projeto

A parceria de indústria realizada com uma das maiores plataformas digitais de delivery de produtos e serviços da América Latina proporcionou uma oportunidade de atuação de alunos, professores e parceiros em um ecossistema caracterizado por sua elevada complexidade tecnológica e operacional.

Foram definidos dois aspectos de *business drivers* norteadores deste estudo: (i) a otimização da operação logística, com foco em estabilidade e previsibilidade sistêmica, e (ii) a saúde do sistema como monitoramento de requisitos não funcionais, acompanhando a estabilidade e confiabilidade da aplicação, ambos avaliados por meio da melhoria contínua na relação de monitoramento e prevenção à falhas do sistema, agregando suporte de engenharia.

Desse modo, o projeto foca em demonstrar a viabilidade e eficácia de implementação de aferições de qualidade atribuindo valor ao código utilizando como base de estudo o aplicativo com serviços digitais para os entregadores, que utilizam bicicletas ou motocicletas, conectados com fornecedores de e-commerce, de onde o material adquirido pelos clientes finais eram vendidos, pagos, empacotados e liberados para o fluxo de retirada e entrega.

3.2 Adição de Valor do Projeto

A plataforma citada hoje tem a sua qualidade mantida por processos convencionais de estágios de desenvolvimento, testes funcionais, e mecanismos de controle de qualidade humana, gerencial, com pouca automação. A tabela 3.2.1 mostra o foco escolhido pela equipe para adicionar valor ao projeto.

Observe-se que na tabela, a adição de valor fica focado nos ganhos que a automação de código traz para o sistema se comparado com a forma atual de controle de qualidade. Nos controles de qualidade atuais, a verificação é baseada em pouca ou nenhuma verificação de da qualidade. Já com o controle de qualidade como código, os cenários de automação já adicionam valor com os códigos que podem verificar se as funcionalidades e as não funcionalidades estão operando de acordo com os níveis esperados.

Tabela 3.2.1 - Adição de valor ao projeto (Fonte:autores)

Fluxo da Plataforma	Controle de Qualidade Atual	Adição de Valor
Fluxo de funcionalidades críticas.	Reativo. Com reclamação de gestores de logística de motos.	Código de testes funcionais; Código de testes não funcionais em volumetrias, acessos simultâneos.
Comportamento do sistema: aspectos não funcionais - tempo de resposta, precisão de saldos, precisão de alocação de corridas.	Reativo. Não verificado em termos de comportamento de sistemas.	Sob volumes e acessos simultâneos: - Códigos de testes de tempos de respostas; - Idem para precisão de saldos de comissões; - Idem para precisão de alocação de corridas.
Ferramentas de controle de qualidade.	Registros em planilhas.	Dashboards digitais com acessos em notebooks e smartphones, com alertas e alarmes.

Fonte: elaborado pelos autores

Os estudantes produziram códigos utilizando algumas práticas realizadas na indústria como a adoção de recursos de BDD (behavior-driven development), conforme ilustra a figura 3.2.1 com um trecho de código que ilustra o controle de qualidade verificando a alocação de entregadores com restrição de tempos, através de um código no padrão Gherkin.

Figura 3.2.1 Trecho de código Gherkin de testes.

alocacao_entregador.feature

```
1   Feature: Busca por entregador disponível
2
3     Scenario: Encontrar um entregador dentro do tempo máximo
4       Given o sistema inicia a busca por um entregador disponível
5       When o sistema encontra um entregador antes de 15 minutos
6       Then o sistema deve alocar o entregador para a entrega
7
```

Fonte: elaborado pelos autores

3.3 Requisitos funcionais e não funcionais considerados

Os requisitos funcionais e não funcionais escolhidos pela equipe foram aquelas que sustentam os direcionadores de negócios da plataforma, que são aquelas que suportam o bom andamento dos trabalhos dos entregadores, dos fornecedores de e-commerce que devem ter as suas mercadorias vendidas entregues com a devida agilidade para os clientes finais. Observe-se que nesta plataforma de entregadores o foco é B2B (Business to Business) porém ao final o benefício é B2C (Business to Consumer), pois o afetado, o envolvido é o cliente que recebe no conforto da sua residência as encomendas de maneira precisa e íntegra.

A tabela 3.3.1 indicam algumas especificações consideradas para serem verificadas como código de aferição da qualidade.

Tabela 3.3.1 - Requisitos selecionados para aferição por código (Fonte: autores).

Classificação	**Fluxo a ser aferido** (Siglas: RF - Requisito Funcional, RNF - Requisito Não Funcional)
RNF	Alocação automática de encomendas considerando raios de proximidade (entregador, fornecedor/estoque, consumidor);
RNF	Atualização em tempo real dos entregadores, com predição de atrasos e entregas nos níveis contratados;
RNF	Taxas de atualização até 1 seg, em 99% das vezes, dos mapas de entregas (normais, em atraso, em alocação);
RNF	Taxa de ganho em comissão atualizado em até 2 segundos, 99% das vezes dos entregadores;
RNF	Tempo de resposta das demais solicitações em até 3 segundos, 95% da demanda;
RF	As funcionalidades de consultas por cada um dos perfis (entregador, fornecedor, gestor de entregas, suporte aos clientes), decorrentes dos demais requisitos não funcionais listados acima.

Fonte: elaborado pelos autores

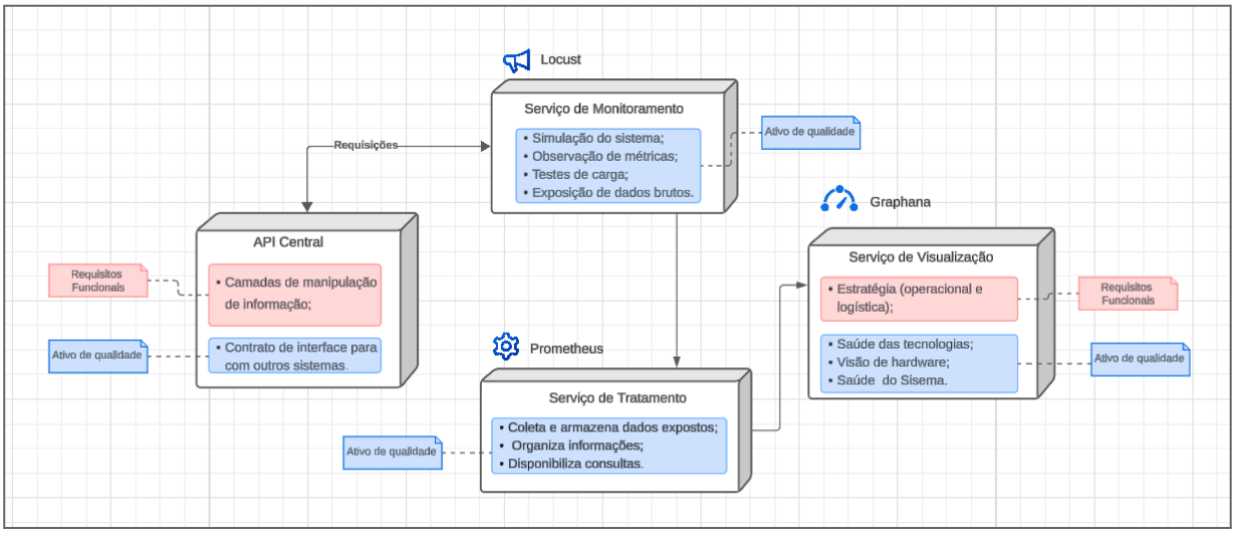
Observe que os itens a serem automatizados garante que a aferição não é manual. Os requisitos funcionais e não funcionais da tabela foram mapeados para serem aferidos por códigos executáveis, diferentemente de processos manuais, humanos. Aos estudantes ficou bem entendido que se os itens da tabela na verdade indicam riscos de

indisponibilidades do sistema e também de erros de alocação que sem automação, esses riscos são tratados no pior cenário: procedimentos reativos, a partir de reclamações feitas pelos canais de atendimento e por gestores e diretamente pelos executivos.

3.4 Solução técnica

Com base na adição de valor proposto, no mapeamento de riscos com base nos elementos funcionais e não funcionais citados, os estudantes puderam chegar ao ponto indicado pela disciplina: não é suficiente cuidar da qualidade apenas com as funcionalidades e suporte. Como cuidar da saúde de um ser humano, um sistema crítico como esse deve ser cuidadosamente examinado nos pontos críticos (“coração, pressão, sinais vitais”) para garantir a fortaleza e o controle dos riscos de indisponibilidade. A equipe de projeto então construiu com base nessas visões conectadas ao business drivers, uma visão de engenharia, de solução técnica, capaz de apoiar a gestão da qualidade deste “corpo humano digital da logística”. Ver a figura 3.4.1.

Figura 3.4.1 - Pontos de conexões vitais do sistema e ferramentas (Fonte: autores)



Fonte: elaborado pelos autores

A API Central é o módulo onde a plataforma realiza os acionamentos dos serviços de consulta, requisições e coleta de dados dos algoritmos inteligentes da retaguarda. A API Central desempenha o papel de núcleo especializado do sistema, concentrando tanto a execução da lógica de negócio quanto a responsabilidade pela exposição de contratos de integração com sistemas externos. Já o Serviço de Monitoramento torna-se essencial para garantir que esses requisitos sejam observados, mensurados e avaliados sob condições reais e simuladas de operação. Esse serviço é integrado diretamente à API Central e responde pela instrumentação ativa da aplicação, realizando a coleta automatizada de métricas como tempo de resposta, throughput, taxa de erro por endpoint, variações de latência e disponibilidade sob carga.

Por meio de execuções controladas com ferramentas como o Locust, o sistema é submetido a cenários diversos que avaliam sua resiliência, estabilidade e escalabilidade, permitindo a antecipação de gargalos e pontos críticos. Serviço de Monitoramento são encaminhados a um componente especializado: o Prometheus, responsável pelo armazenamento estruturado e eficiente das métricas técnicas. Este serviço de tratamento atua como um banco de séries temporais orientado à observabilidade, permitindo que cada ponto de controle definido na arquitetura seja consultado, analisado e correlacionado

de forma granular ao longo do tempo e também para predição, conectando-se com algoritmos inteligentes.

Com base nas integrações realizadas, o Prometheus coleta automaticamente informações expostas por endpoints instrumentados, armazenando indicadores como tempo de resposta por rota, porcentagem de erros, uso de CPU, memória e disponibilidade por serviço. Essas métricas não apenas refletem o comportamento operacional do sistema, mas também documentam a evolução da qualidade como um ativo versionado, possibilitando auditorias, rastreamento de regressões e análises comparativas entre diferentes versões da aplicação.

Ao observar a figura 3.4.1 nota-se o Serviço de Visualização, implementado com o uso do Grafana. Esse componente é responsável por transformar os dados operacionais em interfaces analíticas compreensíveis, oferecendo painéis dinâmicos e personalizáveis que permitem a leitura e a interpretação estratégica dos indicadores de qualidade.

3.5 Implementação e resultados

Qual foi o resultado do foco em código no controle de qualidade deste PBL? Um mapa foi construído com base nos resultados produzidos pela equipe organizados no repositório de projeto (Repositório Inteli, 2025). A tabela 3.5.1 destaca este material.

Tabela 3.5.1 - Mapa de código e valor adicionado pelos estudantes (Fonte: autores).

Qualidade *as Code*	**Artefatos**	**Destaques**
Ativo versionável	Gherkin, YAML, DSLs	Requisitos funcionais e não funcionais como código controlado em repositório com mecanismos de gestão da configuração, de maneira uniforme ao fonte.
Testes	BDD, TDD	Validações automatizadas, para testes de regras de negócios, testes técnicos de integração e de regressão. Rastreabilidade do controle de qualidade baseada em dados.
Documento	Restrições de negócios e técnico	Regras técnicas de integração, regras de negócio, regras de lei, limitações de sistemas internos e externos, na forma de código que evidenciam o comportamento dinâmico do sistema.
Métricas	Observability	A observabilidade do sistema, as evidências e as ações para implementação de controle completo, em malha fechada, com tempos de respostas cibernéticas.
Ciclo de Vida	Automação de esteiras	As esteiras de devsecops instrumentados para a obtenção de continuous integration e continuous deployment (CI/CD). Automação e controle de configuração em escala de equipes distribuídas.

Fonte: elaborado pelos autores

Veja que o foco em controle automatizado da qualidade pode cobrir os diversos aspectos da engenharia de software, tanto em qualidade de produtos como também da qualidade de processos, conforme as indicações de normas como a ISO/IEC 122077, ISO/IEC 25010 e as boas práticas de arquiteturas suportadas por código. Essa modelagem de governança de qualidade deverá contribuir fortemente para a grande adoção de geração de códigos utilizando ferramentas de Inteligência Artificial como as LLMs (Large Language Models).

A avaliação do contexto de controle de qualidade revela uma transformação estrutural profunda promovida pela nova arquitetura. No cenário anterior, a qualidade era tratada de forma reativa, não havia uma visão contínua, distribuída ou automatizada da

qualidade, e os atributos não funcionais raramente eram monitorados em tempo de execução. Ainda mais crítico, a conexão entre o código-fonte e a operação logística concreta era praticamente invisível: falhas percebidas em campo — como lentidão nas telas, valores inconsistentes ou falhas de comunicação entre sistemas — não encontravam representação rastreável no sistema técnico. Essa ausência de visibilidade e rastreabilidade compromete a capacidade de diagnóstico, resposta e aprendizado, criando um abismo entre a engenharia de software e a realidade operacional.

A Tabela 3.5.1 descreve como a "Qualidade como Código" transforma artefatos de desenvolvimento em ativos versionados, como requisitos e testes, integrados ao código-fonte. Essa abordagem automatiza validações e o ciclo de vida via esteiras CI/CD, garantindo rastreabilidade e controle. O resultado é um sistema com observabilidade e controle em malha fechada, promovendo maior qualidade e resiliência.

4 CONSIDERAÇÕES FINAIS

Os resultados obtidos foram bastante relevantes para os estudantes, parceiros, professores e orientadores deste módulo de Controle de Qualidade. Para os estudantes, foram 5 repositórios que se transformaram da seguinte forma: se em módulos anteriores, os alunos produziam muitos documentos escritos, nesse buscaram trazer mais diagramas e menos textos. Já partiram para as visões de arquitetura com a criação de códigos de aferição dinâmica de testes de negócios em termos de volumes, precisão e desempenho sempre focando em limites de negócio, no caso da logística de ecommerce B2B e B2C.

Do lado dos parceiros, ficou bastante evidente a diferença entre as práticas atuais e as provocações e desafios enfrentados e entregues pelos alunos. Dos artefatos entregues, código de testes que normalmente são realizados por processos, com pouca adição de valor e registros manuais se tornaram em gráficos e mensagens de alertas ou de alarmes *on line*, em casos críticos no fluxo da operação e atendimento aos clientes, no caso operadores de entregas, motociclistas e clientes pessoas jurídicas.

Os professores de computação e de orientação, em conjunto, puderam refletir e provocar os alunos sobre posturas e atitude de qualidade, incluindo palestras de gestores de empresas digitais do Brasil e de *Big Techs* da Califórnia para discussões de boas práticas sobre a adoção do contexto de codificar não somente os componentes do sistema, mas todo o processo de gestão de configuração, suporte a gestão de mudanças. Um caso importante foi a confirmação de que novas gerações de profissionais são bem mais afeitas ao ferramental dinâmico como dashboards, mensagens, alertas, e demais mídias que combinam com códigos ao invés de documentos volumosos e passivos (como os .docx, planilhas e diagramas extensos).

A pesquisa, um caso prático de PBL em um curso de Engenharia de Software com um parceiro de logística, tem uma generalização limitada. Apesar dos benefícios observados, faltam comparações e métricas quantitativas para medir o impacto no aprendizado e análise comparativas. Pesquisas futuras poderiam replicar o modelo em outros contextos, realizar estudos comparativos e desenvolver instrumentos quantitativos para avaliar o impacto da "Qualidade como Código".

AGRADECIMENTOS

Agradecimento ao Instituto Inteli (direção e coordenação) do curso de engenharia de software pelo estímulo ao compartilhamento de informações e conhecimentos relacionados ao ensino PBL e os impactos no conhecimento dos alunos, especialmente sobre os aspectos de controle de qualidade de sistemas tanto em posturas técnicas, como em liderança. Aspectos de qualidade de arquitetura são difíceis de serem assimilados quando se trata de requisitos não funcionais que trazem precisão, resiliência, disponibilidade difíceis de serem medidos e controlados pelas sutis relações de *tradeoffs*.

REFERÊNCIAS

Repositório Inteli. **Aplicação, Testes, Ferramentas Utilizadas.** Solicitar acessando o site do Instituto Inteli https://www.inteli.edu.br/. Acessado em Maio de 2025.

ARAKAKI, R.; HAYASHI, V. T. and RUGGIERO, W. V. **"Available and Fault Tolerant IoT System: Applying Quality Engineering Method"**, 2020 International Conference on Electrical, Communication, and Computer Engineering (ICECCE), 2020, pp. 1-6, doi: 10.1109/ICECCE49384.2020.9179341.

ARAKAKI, R. et al. **Avaliação do oferecimento a distância de laboratório de eletrônica digital por meio de objetivos de aprendizagem e métricas do AVA.** In: ANAIS do XLIX Congresso Brasileiro de Educação em Engenharia, 2021. DOI: https://doi.org/10.37702/cobenge.2021.3566;

Barrows, H. S. (1996). **Problem-Based Learning in Medicine and Beyond: A Brief Overview. New Directions for Teaching and Learning**, 1996, 3-12. http://dx.doi.org/10.1002/tl.37219966804;

BASS, Len; CLEMENTS, Paul; KAZMAN, Rick. **Software Architecture in Practice**. 3. ed. Boston: Addison-Wesley, 2012.

CRUZ, José Jaime da. **Entendendo e ajustando malhas de controle**. São Paulo: GAESI – EPUSP, 2004.

CUGNASCA, Paulo Sérgio *et al.* **O Papel do Laboratório de Eletrônica Digital como Disciplina Integradora de Competências para o Desenvolvimento de Projetos de Engenharia.** *In:* 52º Congresso Brasileiro de Educação em Engenharia, 2024, Vitória. **Anais**. Vitória. DOI: 10.37702/2175-957X.COBENGE.2024.5074.

HAYASHI, V. et al. **Laboratório Remoto para o Ensino de Engenharia.** In: S WORKSHOPS DO CONGRESSO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO (WCBIE), 9. , 2020, Online. Anais [...]. Porto Alegre: Sociedade Brasileira de Computação, 2020. p. 187-194. DOI: https://doi.org/10.5753/cbie.wcbie.2020.187;

HAYASHI, V. T. et al. **Laboratório Virtual com Dados Reais.** In: SIMPÓSIO BRASILEIRO DE EDUCAÇÃO EM COMPUTAÇÃO (EDUCOMP), 1. , 2021, On-line. Anais [...]. Porto Alegre: Sociedade Brasileira de Computação, 2021 . p. 296-304. DOI: https://doi.org/10.5753/educomp.2021.14497;

HAYASHI, Victor Takashi et al. **Implementation of PjBL With Remote Lab Enhances the Professional Skills of Engineering Students.** August 2023. IEEE Transactions on Education PP(99):1-10. DOI:10.1109/TE.2023.3243532;

INTELI PPC. **Projeto Pedagógico do Curso de Bacharelado em Engenharia de Software.** 2024. Acesso em Janeiro de 2025.

ISO/IEC 12207:2008 **Systems and software engineering – Software life cycle processes** http://www.iso.org/iso;

ISO/IEC 25010:2011 - **Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models** http://www.iso.org/iso/catalogue_detail.htm?csnumber=35733;

ISO/IEC/IEEE 42010:2022. **Systems and software engineering - Architecture description** https://www.iso.org/standard/74393.html;

ISO/IEC 10746-2:2009 **Information technology — Open distributed processing — Reference model: Foundations;**

INSTITUTO NACIONAL DE PADRÕES E TECNOLOGIA – NIST; DEPARTMENT OF DEFENSE – DoD. **Guide to Software Architecture Evaluation**. Gaithersburg: NIST/DoD, 2011. Disponível em: https://www.nist.gov. Acesso em: 26 mai. 2025;

PRESSMAN, ROGER; MAXIM, BRUCE; Revisão Técnica ARAKAKI, R. **Engenharia de Software: Uma Abordagem Profissional** [Versão em Português Brasil]. AMGH 2021;

SEI – Software Engineering Institute. **ATAM: Architecture Tradeoff Analysis Method**. Pittsburgh, PA: Carnegie Mellon University, 2007. Disponível em: https://sei.cmu.edu. Acesso em: 26 mai. 2025.

HAYASHI, V.; GARCIA, V.; ANDRADE, R. A.; ARAKAKI, R. (2020). **OKIoT Open Knowledge IoT Project: Smart Home Case Studies of Short-term Course and Software Residency Capstone Project.** In: Proceedings of the 5th International Conference on Internet of Things, Big Data and Security - IoTBDS, ISBN 978-989-758-426-8; ISSN 2184-4976, pages 235-242. DOI: 10.5220/0009366002350242.

HAYASHI,V. T.; FUJII, T.; ARAKAKI, R.; AMARAL, H.L.M.; SOUZA, A.N.; 2020. **Boa Energia: Base de Dados Pública de Consumo Residencial com Qualidade de Dados**. Evento: XXXVIII Simpósio Brasileiro de Telecomunicações e Processamento de Sinais (SBrT2020); DOI: 10.14209/SBRT.2020.1570648906.

A PROJECT-BASED LEARNING CASE STUDY FOR SOFTWARE ENGINEERING: QUALITY CONTROL ADDING VALUE BY CODING - GETTING DOCUMENTATION AND QUALITY ASSURANCE AS CODE.

Abstract: *This work describes how students encounter challenges in ensuring the quality of software systems by implementing quality control procedures. At first, students and some professionals think these activities are boring: documenting and applying test cases. No one likes this kind of activity. Professors and coaches created one dimension of challenges for students: code quality, code documentation, code control, and code rules. From a business view and technology viewpoint, everything is code. As a result, the system code created for quality control and quality assurance enabled students and business partners to develop a significant project related to the Quality Control PBL Module.*

Keywords: *Architecture as code, Documentation as code, Functional and Non Functional requirements, PBL Project Based Learning, Software Engineering, Quality Assurance, Quality Control, Business Drivers, ATAM, ArchitectureTradeoff.*