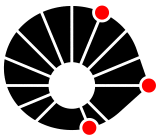

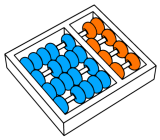

Responsabilidades Arquiteturais para Experimentação Online

Chan Kyung Kim *Breno Bernard Nicolau de França*

Relatório Técnico - IC-PFG-19-15
Projeto Final de Graduação
2019 - Julho

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

Responsabilidades Arquiteturais para Experimentação Online

Chan Kyung Kim* Breno Bernard Nicolau de França*

Resumo

Muitas empresas na indústria tiveram que se adaptar para oferecer produtos e serviços através de *software*. De forma geral, mesmo um *software* já lançado pode ser constantemente melhorado através da Experimentação Online, também conhecido como testes A/B, amplamente aplicado no mundo. Porém, para que uma empresa toma decisões *data-driven* para a melhoria do software é importante que exista uma abordagem sistemática para a Experimentação. Temos como objetivo propor responsabilidades arquiteturais essenciais que consigam suportar o desenvolvimento, implementação e avaliação dos experimentos, além de ser capaz de coletar e analisar dados. Para isso, este trabalho analisa quatro arquiteturas diferentes da academia, identificando pontos em comum, além de explorar as três ferramentas mais utilizadas hoje na indústria para testes A/B. A contribuição deste relatório é uma proposta de uma arquitetura com componentes essenciais para Experimentos Online. Vemos, também, que a indústria e a academia se relacionam bastante neste tema.

*Instituto de Computação, Universidade Estadual de Campinas, 13081-970 Campinas, SP.

Sumário

1 Introdução

Com o crescimento acelerado da digitalização na indústria, diversos setores tiveram que se adaptar para alavancar suas vendas, aumentando assim o número de empresas que se tornaram provedoras de produtos e serviços por meio do uso de *software* [1]. Tanto empresas antigas, quanto novas empresas tiveram que se encaixar a essa nova realidade.

Mesmo que um *software* já tenha sido lançado, existem diversas flexibilidades que permitem que o produto inicial seja melhorado [1]. Uma das possíveis flexibilidades é a Experimentação Online, conhecida também como Testes A/B.

Em síntese, Testes A/B consistem em comparar uma variante A (controle), com uma variante B (tratamento), dividindo aleatoriamente os usuários entre as duas variantes, com o objetivo de melhorar uma métrica específica [2], [10], como mostrado na Figura 1 (adaptada do artigo [2]).

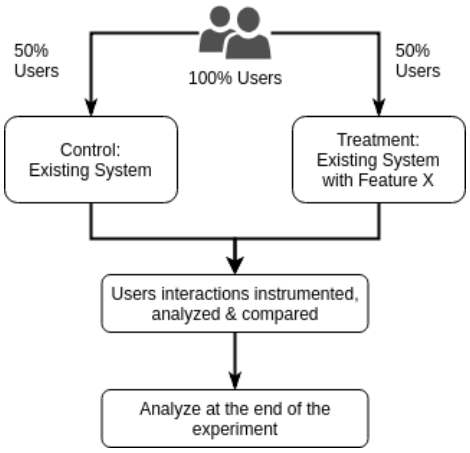

Figura 1: Fluxo high-level de uma experimentação online

A implementação da Experimentação Online traz um bom *return-on-investment* (ROI) e, com uma infraestrutura apropriada pode acelerar a inovação [2]. Além disso, quando bem conduzidos, de forma sistemática, elas trazem um entendimento mais preciso do que os clientes priorizam e provém provas para decidir sobre uma métrica chave para a mudança específica de um produto [3]. Ou seja, as decisões tornam-se *data-driven*, escutando os clientes, em vez de serem baseados no *HiPPO* (Highest Paid Person's Opinion), opiniões de desenvolvedores ou gerentes. [4][7][9].

Assim, uma empresa consegue impulsionar inovação por meio da experimentação. Isso faz com que os custos de testes e falhas experimentais tornem-se pequenos, auxiliando com propostas de novas ideias a serem implementadas, fazendo com que as empresas tenham seu sucesso alavancado. Diversos artigos mostram melhoria nos produtos/serviços por meio dos benefícios provenientes dos experimentos online. Tanto na academia quanto na indústria, em empresas como Microsoft, Google, Facebook, entre outras [1][3][5][6][8], foi possível observar resultados expressivos.

O desenvolvimento de novas features deve ser feito, idealmente, de forma sistemática

e *data-driven* [7]. Dado isso, para que os experimentos onlines sejam devidamente implementados, é necessário uma arquitetura que suporte tanto o desenvolvimento, quanto a implementação e avaliação dos experimentos, junto com uma infraestrutura capaz de coletar e analisar os dados, utilizando *feedbacks* de usuários reais em grande escala [9].

2 Justificativa

Empresas que estão trilhando em direção ao desenvolvimento baseado em experimentos, enfrentam diversos desafios. O custo e consumo de tempo elevados, diferentes times que podem levar a objetivos conflitantes e, falta de uma abordagem sistemática para rodar experimentos em diferentes domínios, leva as empresas a criarem diversas plataformas de experimentos [2][7].

[1] também afirma que ainda não existe um *framework* detalhado para realizar desenvolvimento de software baseado em experimentação de forma sistemática. É necessário um *framework* que englobe uma infraestrutura técnica do produto; um processo de desenvolvimento do *software*; requerimentos acerca de competências que os desenvolvedores precisam para projetar, executar, analisar e interpretar experimentos; e os recursos que a empresa precisa para operar e gerenciar baseada experimentação.

Além disso, em [11] vemos que, apesar de muitas empresas se interessarem pela experimentação online, nem todas conseguem a incorporar aos negócios. Entre as principais dificuldades (Figura 2, adaptada do artigo [11]), são citados como principais causas: problemas com arquitetura, falta de expertise, base de usuários pequena e falta de interesse em investir por motivos comerciais.

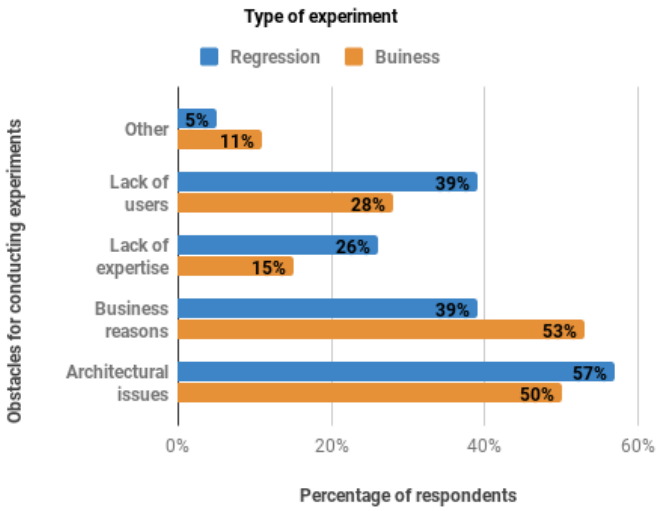

Figura 2: Principais obstáculos para implementação de experimentação

3 Objetivo e Método

3.1 Objetivo

Esse trabalho tem como objetivo explorar as arquiteturas e ferramentas existentes hoje na academia e plataformas de experimentação (*Optimizely, Google Optimize e VWO*) na indústria para propor responsabilidades arquiteturais essenciais para a execução de experimentos online e ver como elas se relacionam com as plataformas online.

3.2 Revisão bibliográfica de artigos acadêmicos

A princípio, foi realizada uma pesquisa na literatura técnico-científica para levantar artigos que abordassem sobre o tema "Experimentação Online" e, especificamente, sobre aspectos relacionados à arquitetura. Foram utilizados, na ferramenta Google Scholar, os termos "*online controlled experiments*", "*architecture continuous experimentation*"e"*architecture innovation experiment*".

Para este trabalho, foram considerados os artigos [1], [5], [7] e [9], e suas arquiteturas. A seguir, seguem os detalhes de cada arquitetura apresentada nos artigos e suas principais contribuições.

3.3 Compilação das responsabilidades

Revisando os artigos, percebe-se que há uma diferença em granularidade dos detalhes de cada arquitetura, assim, foram escolhidas as responsabilidades de uma forma que fosse mais detalhada, dentro do conjunto de responsabilidades dos quatro artigos. Feito isso, foram mapeadas os componentes menos detalhados que abrangiam mais funções em múltiplas responsabilidades. Foram considerados as responsabilidades que foram explicitamente citados nos artigos.

Além disso, os nomes de cada componente foram mantidos para cada artigo e compilados em uma tabela a ser mostrada na seção 5. As descrições de cada responsabilidade foram feitas de uma forma a abranger os quatro artigos. Especificamente, para a arquitetura RIGHT, no lugar de funções de trabalho, as responsabilidades foram adaptadas para refletirem os componentes da arquitetura.

3.4 Levantamento das ferramentas existentes na web

Por conta da grande quantidade de ferramentas presentes hoje na *web* para experimentos online, além de diversas funcionalidades que cada um ferramentas consegue oferecer, foi feita uma análise das ferramentas mais utilizadas. Os critérios de seleção foram:

- Ferramentas que rodam testes A/B
- Os top 100 mil sites em tráfego que utilizam plataformas de experimentação online

Com a ajuda de alguns sites [12], [13], [14] que medem o *market share* da utilização das ferramentas, foram levantadas as três que são utilizadas no mercado e muito citadas por sites de indicação de melhores ferramentas de testes A/B [18][19][]20][21][22]. São elas:

- *Google Optimize*
- *Optimizely*
- *VWO - Visual Website Optimizer*

Como podemos ver na Figura 3, todas as 3 ferramentas aparecem nos top 3 em market share em utilização, seguindo os critérios citados acima. Conseguimos afirmar que para o critério utilizado, as três ferramentas possuem um market share significativo de mais de 60%. Para o market share da SimilarTech foram consideradas apenas as 8 top ferramentas.

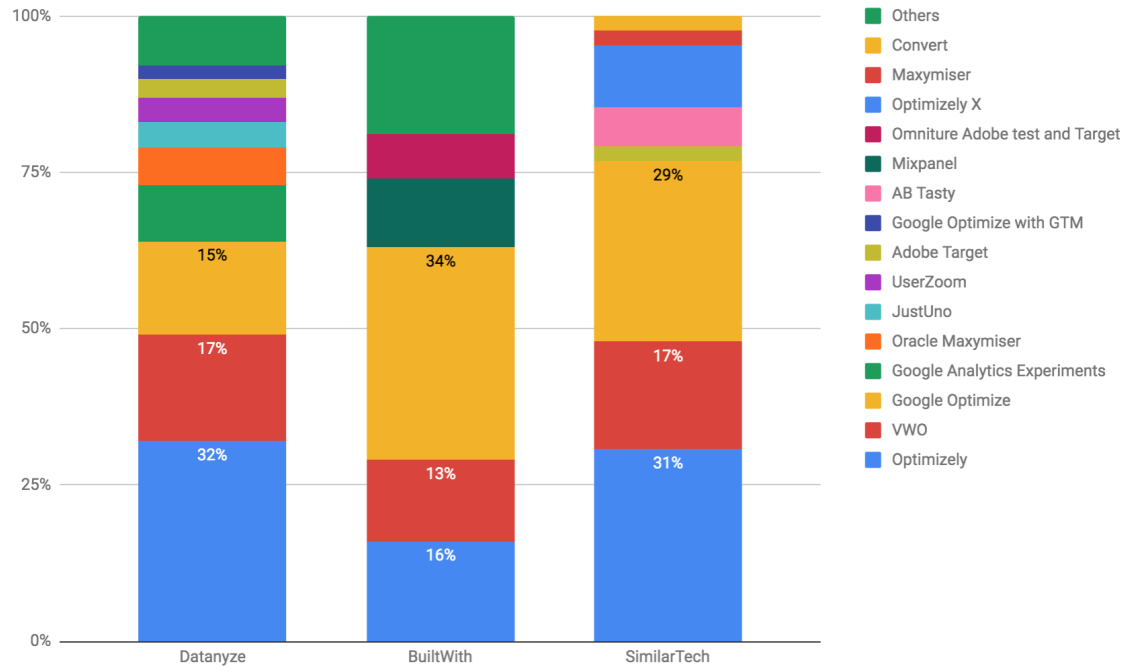

Figura 3: *Market share* Datanyze, BuiltWith e SimilarTech

3.5 Mapeamento de funcionalidades das ferramentas

Foram pesquisadas funcionalidades em cada ferramenta que se relacionassem com as responsabilidades já mapeadas pelos artigos. Para isso, foram exploradas os sites de cada ferramenta ([15], [16], [17]), coletando as funcionalidades que cada uma oferecia, pesquisando na área de *resources* e, além disso, foi feita uma pesquisa de casos de usos exemplos e reais utilizando as ferramentas passo-a-passo. Paralelamente, foi feita uma comparação com as responsabilidades dos artigos e fazendo um *check* nos itens que as ferramentas possuíam.

4 Trabalhos Relacionados

4.1 The RIGHT model for Continuous Experimentation

Neste artigo, Fagerholm [1] tem como objetivo examinar as pré-condições para estabelecer um sistema de experimentação para experimentos contínuos com clientes, respondendo a pergunta: Como a Experimentação Contínua de produtos e serviços altamente dependentes de software pode ser organizada de forma sistemática? Em seguida, a pergunta é dividida em duas:

- Qual é o modelo de processo adequado para Experimentação Contínua de produtos e serviços altamente dependentes de software?
- Qual é o infraestrutura da arquitetura adequada para Experimentação Contínua de produtos e serviços altamente dependentes de software?

No que diz respeito à parte arquitetural do trabalho, o autor propõe um modelo de Experimentação Contínua chamada RIGHT (*Rapid Iterative value creation Gained through High-frequency Testing*) que possui a infraestrutura de arquitetura apresentada na Figura 4, adaptada do artigo [1].

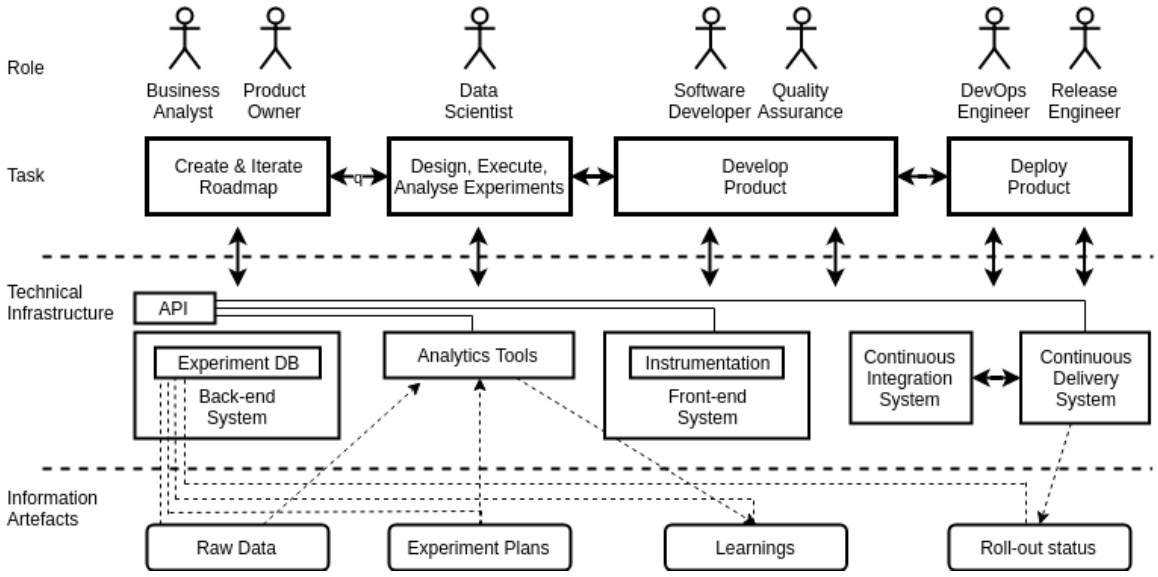

Figura 4: Infraestrutura de arquitetura do modelo RIGHT

- Business Analyst ou Product Owner, ou a equipe de Product management
  - Responsáveis para a criação e iteração dos roteiros dos experimentos;
  - Consultam planos, resultados e aprendizados de experimentos existentes;
  - Trabalham com o Data Scientist para comunicar as suposições do roteiro e as áreas que precisam ser testadas.

- Data Scientist
  - Projetam, executam e analisam os experimentos;
  - Acessam raw datas pelo back-end system, acessam planos dos experimentos e realizam as análises;
  - Armazenam os resultados produzidos em forma de aprendizados no back-end system;
  - Se comunicam com os Developers e Quality Assurance.
- Developer e Quality Assurance
  - Cuidam do desenvolvimento do MVP (Minimum Viable Product)/MVF (Minimum Viable Feature) e do produto final;
  - Cuidam da instrumentação apropriada para o front-end system, parte que será mostrada ao usuário;
  - Trabalham para desenvolver e otimizar as features necessárias e submeter para produção;
  - MVP/MVF/Produtos finais são implementados através do Continuous Integration System e Continuous Delivery System.
- DevOps
  - Fazem a ponte entre o time de desenvolvimento e operações.
- Release Engineer
  - Supervisionam e gerenciam distribuições em produção.

Uma das principais funções do Continuous Delivery System é a informação do status do r*oll-out* do software que ele provém, permitindo a outras funções monitorar a execução do experimento, e obtendo, por exemplo, as condições em que um *software* foi implementado e o *response rate*.

Apesar de bastante detalhado, o autor ainda expõe a arquitetura de uma forma não exaustiva e alto-nível e afirma que várias funções relacionadas a operações foram omitidas, por exemplo, *reliability engineer*. Além disso, foram omitidas quais informações deveriam ser visíveis para quais funções.

4.2 Online controlled experiments at large scale

Neste artigo, Kohavi [5] implementa uma sistema de experimentação online na Microsoft, especificamente na Bing, chamada *Bing Experimentation System*, que ajudou a trazer resultados expressivos para a empresa rodando mais de 200 experimentos ao mesmo tempo. A arquitetura do sistema é dividida em 3 grandes partes, como mostrada na Figura 5, adaptada do artigo [5]:

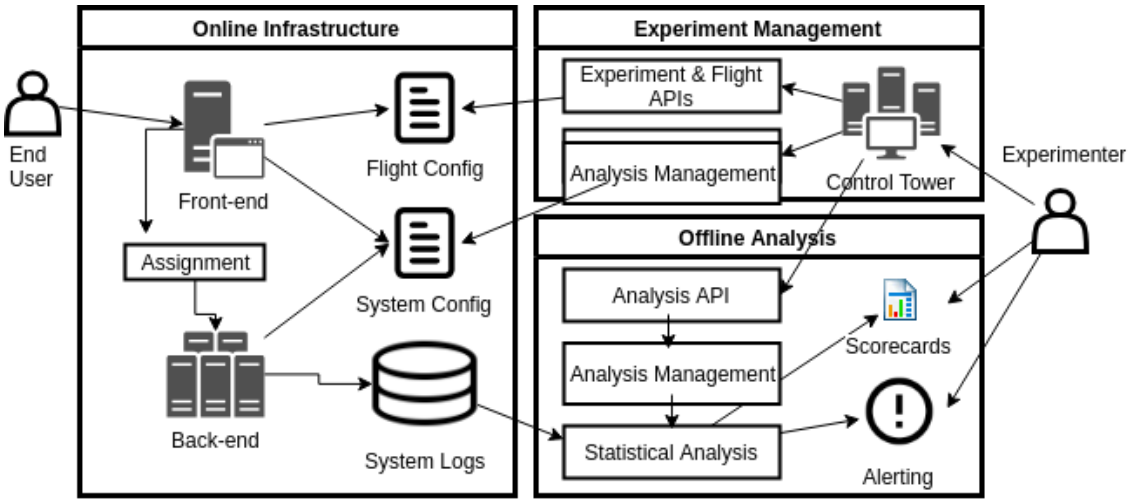

Figura 5: Arquitetura do Bing Experimentation System

- Online Infrastructure:
  - Os servidores *frontend* realizam o *assignment* cada requisição aos diversos flights (como são chamadas as variantes na Bing) à medida que são recebidas;
  - É feita uma pseudo randomização dos *user id* para garantir consistência nos assignments;
  - Informações de *logs* do sistema, incluindo as assignments das variantes, são guardadas para serem processadas e analisadas *offline*.
- Experiment Management:
  - É o responsável pelo *Control Tower* para gerenciar os experimentos;
  - Utiliza APIs para definir e executar experimentos;
  - Configuration API permite aos desenvolvedores a criarem as configurações de um experimento.
- Offline Analysis:
  - É gerada um *scorecard*, um resumo do experimento, pelo fluxo de análise offline do experimento, que gerencia grandes quantidades de dados e experimentos;
  - *Scorecards* são utilizados para acompanhar os impactos das mudanças do experimento ao longo do tempo;
  - Gerencia e otimiza o fluxo de execução de múltiplas análises utilizando os *logs*;
  - Monitora e alerta os sistema, detectando automaticamente tanto a qualidade dos dados quanto um evento de impacto negativo aos usuários.

4.3 Your System Gets Better Every Day You Use It: Towards Automated Continuous Experimentation

Neste artigo, Mattos, Bosch e Olsson [7] realizaram uma análise de diversas arquiteturas e suas qualidades relacionadas a sistemas auto-adaptativas e, em seguida, desenvolvem um *framework* de arquitetura que suporta experimentação contínua automatizada baseado nessa análise. Tal arquitetura, mostrada na Figura 6 adaptada do artigo [7], é composta pelos seguintes componentes, podendo-se retirar um ou mais componentes para alguma aplicação específica:

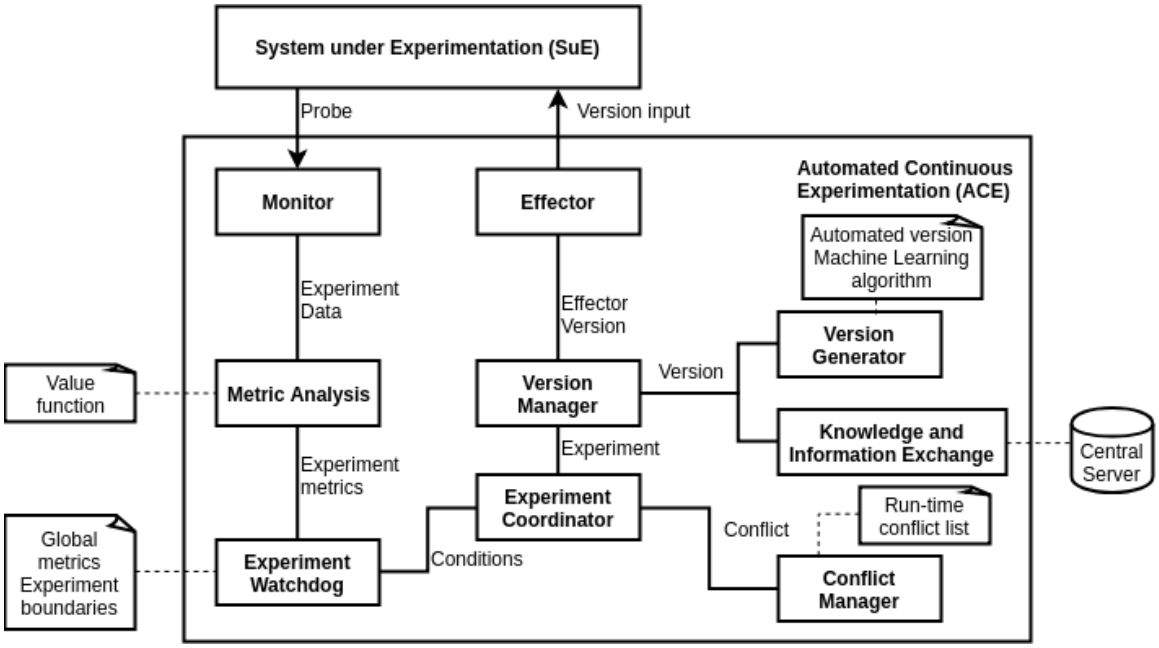

Figura 6: Framework da arquitetura da Experimentação Contínua Automatizada

- Monitor
  - Responsável pela coleta de dados locais e globais, vindos do SuE (*System under Experimentation*).
- Effector
  - Responsável pela interface com o sistema gerenciado.
- Experiment coordinator
  - Responsável por rodar o experimento e coordenar com o Version Manager;
  - Fazer Testes A/A (*sanity check*), A/B/n (experimentos com mais de uma variante de tratamento) e explorar experimentos cruzados;

  - Monitora quando e se um experimento deve rodar, número de experimentos que devem ser realizados e quais soluções são mais significantes;
  - Recebe inputs do Conflict-list Manager e Experiment Watchdog.

- Version Manager
  - Gerenciar e gerar diferentes versões para os experimentos, mudando parâmetros e variantes;
  - Mantém uma lista das versões já usadas e recebe inputs do Knowledge Exchange e Version Generator.

- Version Generator
  - Adapta diferentes algoritmos de inteligência artificial para os testes.

- Experiment Watchdog
  - Checa as condições para rodar um experimento, como: quando um experimento deve continuar ou parar;
  - Para um experimento se o mesmo passar dos limites pré-definidos ou está prejudicando alguma métrica global.

- Conflict-list manager
  - Monitora quais componentes estão em experimentação e quais fatores elas afetam.

- Metric Analysis
  - Monitorar o comportamento do experimento;
  - Rodar análises estatísticas e realizar um reflexão da qualidade do desempenho.

- Knowledge and Information Exchange
  - Responsável por compartilhar as soluções descobertas no processo de experimentação e aprender sobre as soluções validadas.

4.4 Architecture for Large-Scale Innovation Experiment Systems

Eklund e Bosch [9] procuram, neste artigo, responder a seguinte pergunta: Quais são os princípios de uma arquitetura de software para concretizar um sistema de experimentos de inovação em larga escala para sistemas com software embarcado?

A seguir, os autores propõem uma arquitetura mostrada na Figura 7, adaptada do artigo [9], que suporta a implementação de experimentação de múltiplas partes do software e que controla de forma autônoma quando um teste deve ser rodado, permitindo até testes A/B locais. Leituras e análises são feitas em tempo real e de forma embarcada. A arquitetura consiste nas seguintes partes:

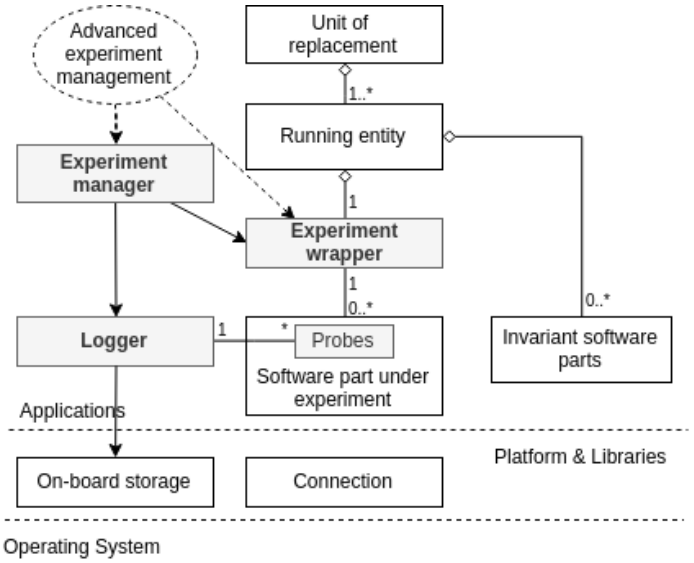

Figura 7: Arquitetura de gerenciamento dos experimentos

- Experiment manager:
  - Responsável por decidir quais experimentos serão executados, rodar o experimento e juntar e analisar os dados coletados.
- Experiment wrapper:
  - Permite ao Manager decidir em tempo de execução quais experimentos rodar, permitindo o teste A/B ou revertendo para *default*.
- Probe:
  - Parte do software que está sendo testado.
- Logger e On-board storage:
  - Guarda os dados coletados e analisados providos pela sistema embarcado para serem *uploaded* para análises futuras.
- Unit of replacement:
  - Cuida tanto da parte de implementação quanto do versionamento do *software*

Os autores também propõem incluir um mecanismo para mitigação de riscos para experimentos críticos para a segurança. Tal componente, mostrado na Figura 8 adaptada do artigo, seria responsável por:

- Avaliar se a aplicação está rodando em limites seguros;

- Controlar e configurar a operação para rodar em limites seguros.

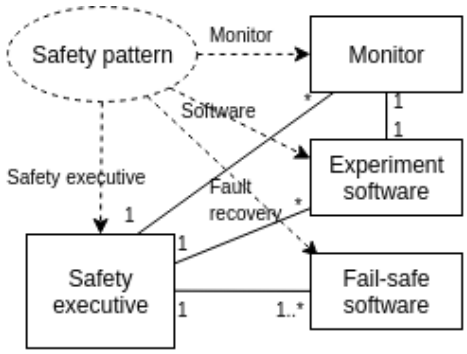

Figura 8: Safety Pattern

5 Resultados

5.1 Síntese de Responsabilidades

Uma vez mapeadas todos os componentes das arquiteturas dos artigos, foi feita uma análise das responsabilidades e foram compiladas de forma exaustiva na Tabela 1. Para cada artigo, foram colocados os nomes de cada componente de forma a refletir o artigo e para facilitar futuras consultas.

#	**Responsabilidade do componente**	[1]	[5]	[7]	[9]
1	Coleta de dados	Backend system	Online Infrastructure	Monitor	Logger
2	Análise dos dados coletados	Analytics Tools	Offline Analysis	Metric Analysis	Experiment manager
3	Interface com o sistema em experimento	Instrumentation (Frontend system)	Online Infrastructure	Effector	-
4	Rodar o experimento	Backend system/ Analytics Tools	Experiment Management	Experiment coordinator	Experiment manager
5	Segmentação de usuários	-	Online Infrastructure	-	-
6	Gerenciar e/ou gerar diferentes versões de experimentos	-	Experiment Management	Version Manager	Experiment manager
7	Uso de algoritmos de IA para os experimentos	-	-	Version Generator	-
8	Determinar as condições para rodar os experimentos	Experiment DB (Backend system)	Offline Analysis	Experiment Watchdog	Safety Pattern
9	Monitorar os componentes/variantes que estão em experimentação	-	-	Conflict-list Manager	Experiment wrapper
10	Monitorar o comportamento do sistema em experimento e avaliar seu desempenho	Continuous Delivery System	Offline Analysis	Metric Analysis	-
11	Compartilhar soluções descobertas e lições aprendidas	Backend system	-	Knowledge and Information Exchange	-
12	Implementação para produção	Continuous Integration/ Delivery System	-	-	Unit of replacement

Tabela 1: Responsabilidades e componentes dos artigos

A seguir, temos as descrições de cada responsabilidade citada na Tabela 1:

1. Dados locais e globais provenientes do experimento são coletados
2. É feita a análise dos dados previamente coletados, podendo ser já processados
3. É um ponto de contato com o experimento, onde há uma interface para interação com o experimento
4. Experimentos são realizadas através de testes A/A (*sanity checks*), A/B/n (experimentos com mais de uma variante tratamento) ou experimentos cruzados.
5. É feita uma divisão de quais usuários recebem quais variantes e quais experimentos de uma forma aleatória e consistente
6. Experimentos são gerenciados podendo ser gerados ou modificados, mudando parâmetros e variantes

7. São utilizados algoritmos de IA para experimentação
8. Define se um sistema deve continuar um experimento e quando deve parar. Pode parar um experimento caso saia dos limites estabelecidos ou esteja danificando alguma métrica global.
9. Monitoramento em tempo real dos componentes/variantes sendo testados e quais fatores as afetam, de forma a evitar confusão entre experimentos
10. Monitoramento do experimento, fazendo avaliação do desempenho e análises estatísticas
11. Compartilhamento das soluções descobertas e lições aprendidas através de uma infraestrutura interna
12. Implementação da variante vencedora para produção

5.2 Ferramentas para Experimentação Online

Tendo sido mapeadas as responsabilidades, foram feitas os *checks* com as funcionalidades das ferramentas online e compilados na Tabela 2, em que "X" representa que tal ferramenta possui a funcionalidade e "*", que possui parcialmente.

#	**Responsabilidade do componente**	**Optimizely**	**Google Optimize**	**VWO**
1	Coleta de dados	x	x	x
2	Análise dos dados coletados	*	x	x
3	Interface com o sistema em experimento	x	x	x
4	Rodar o experimento	x	x	x
5	Segmentação de usuários	x	x	x
6	Gerenciar e/ou gerar diferentes versões de experimentos	x	x	x
7	Uso de algoritmos de IA para os experimentos			
8	Determinar as condições para rodar os experimentos	x	x	x
9	Monitorar os componentes/variantes que estão em experimentação			
10	Monitorar o comportamento do sistema em experimento e avaliar seu desempenho	x	x	x
11	Compartilhar soluções descobertas e lições aprendidas	*	*	*
12	Implementação para produção			

Tabela 2: Responsabilidades e funcionalidades das ferramentas

Podemos observar que as ferramentas possuem basicamente as mesmas funcionalidades, tanto pela questão técnica quanto pela questão competitiva, em que uma ferramenta que

não oferece alguma funcionalidade básica ou geral, pode perder ou até mesmo não conseguir atrair novos clientes.

Por exemplo uma função em comum que todas as ferramentas possuem é a interface com o sistema de experimentação, em que o experimentador pode configurar um experimento visualmente, sem ser programaticamente, através de um editor WYSIWYG (*What You See Is What You Get*), modificando componentes dentro de uma página web ou aplicativo.

No caso da análise de dados coletados para o *Optimizely*, a ferramenta permite integração com outras ferramentas como o *Google Analytics*, porém não possui uma plataforma própria para análise dos dados. Para o compartilhamento das soluções, todas ferramentas mantém um histórico dos testes feitos que ficam armazenados de forma organizada, porém seria necessário um time para que o compartilhamento fosse feito de forma ativa para o conhecimento das pessoas e grupos de pessoas interessadas.

As funcionalidades que não são cobertas pelas ferramentas devem ser incorporadas pelo time de desenvolvimento para que, por exemplo, não tenham conflitos entre experimentos ou para implementar a solução vencedora para a produção.

5.3 Responsabilidades arquiteturais essenciais

Fazendo uma análise das tabelas, foram consideradas essenciais as seguintes responsabilidades: 1, 2, 3, 4, 6, 8, 10 e 12, sendo estas encontradas em pelo menos 3 das 4 arquiteturas analisadas. Algumas razões justificam a não-inclusão das responsabilidades não consideradas essenciais:

- 5: A segmentação de usuários, apesar de importante para a consistência dos dados coletados, não impedem de um experimento rodar. Apesar disso, podemos observar que todas as ferramentas possuem essa funcionalidade.
- 7: Apesar de algoritmos de IA serem um adicional com grande potencial, elas não são necessárias para rodar um experimento. Nenhuma ferramenta permite essa funcionalidade.
- 9: Apesar de bastante importante uma equipe tomar cuidado para que dois experimentos não se interfiram, experimentos criados com certo cuidado e senso crítico podem evitar possíveis colisões entre variantes. Também não é um bloqueio para rodar um experimento
- 11: Compartilhar soluções podem ser bastante úteis quando feito de forma estruturada, porém também não impedem de um experimento rodar

A única exceção entre os componentes que não estão em 3 das 4 das arquiteturas, é o 12. Pode ser considerada essencial que, após um experimento bem sucedido, a variante vencedora seja implementada para que tenha impacto na métrica estabelecida pela empresa.

6 Conclusão e Trabalhos Futuros

Há uma grande importância em implementar um Experimento Online de forma sistemática capaz de gerar decisões *data-driven*. E para que isso ocorra, é necessária uma arquitetura

que suporte tanto o desenvolvimento, quanto a implementação e avaliação dos experimentos, junto com uma infraestrutura capaz de coletar e analisar os dados.

Neste trabalho, analisando as arquiteturas presentes na literatura, vimos que existem muitos componentes em comum que são essenciais para o desenvolvimento de um Experimento Online. Paralelamente, as ferramentas que encontramos hoje na internet se relacionam bastante com as arquiteturas apresentadas, mostrando que, de certa forma, a academia e a indústria possuem bastante pontos em comum.

Por se tratar de um tema mais específico, onde ainda existem poucos artigos falando especificamente sobre o assunto, podemos pensar como próximos passos, a implementação da arquitetura com os componentes essenciais num projeto real a fim de verificar a funcionalidade da arquitetura. Um outro projeto que pode se extender, seria o entendimento em como as reponsabilidades levantadas neste trabalho são tratadas nas empresas, fazendo pesquisas diretamente com equipes que trabalham com isso nas empresas por meio de entrevistas e coleta de dados.

Referências

[1] FAGERHOLM, F. et al. The RIGHT model for Continuous Experimentation. *Journal Of Systems And Software*, v. 12 3, p.292-305, jan. 2017.

[2] KOHAVI, R. et al. Controlled experiments on the web: survey and practical guide. *Data Mining And Knowledge Discovery*, v. 18, n. 1, p.140-181, 30 jul. 2008.

[3] FABIJAN, A. et al. The Benefits of Controlled Experimentation at Scale. *2017 43rd Euromicro Conference On Software Engineering And Advanced Applications (seaa)*, p.18-26, ago. 2017.

[4] *ExP - Experimentation Platform: Accelerating software innovation through trustworthy experimentation.* Disponível em https://exp-platform.com/hippo/ [acesso em 07/2019].

[5] KOHAVI, R. et al. Online controlled experiments at large scale. *Proceedings Of The 19th Acm Sigkdd International Conference On Knowledge Discovery And Data Mining - Kdd '13*, p.1-9, 2013

[6] KOHAVI, R. et al. Seven rules of thumb for web site experimenters. *Proceedings Of The 20th Acm Sigkdd International Conference On Knowledge Discovery And Data Mining - Kdd '14,* p.1-11, 2014.

[7] MATTOS, D. I.; BOSCH, J.; OLSSON, H. H. Your System Gets Better Every Day You Use It: Towards Automated Continuous Experimentation. *2017 43rd Euromicro Conference On Software Engineering And Advanced Applications (seaa),* p.256-265, ago. 2017

[8] KOHAVI, Ron et al. Online Experimentation at Microsoft. *Third Workshop On Data Mining Case Studies And Practice Prize*, p.1-16, 2009.

[9] EKLUND, U.; BOSCH, J. Architecture for Large-Scale Innovation Experiment Systems. 2*012 Joint Working Conference On Software Architecture & 6th European Conference On Software Architecture*, Sweden, p.244-248, 2012.

[10] KOHAVI, Ron; LONGBOTHAM, Roger. Online Controlled Experiments and A/B Testing. *Encyclopedia Of Machine Learning And Data Mining,* [s.l.], p.33-48, 2017.

[11] SCHERMANN, Gerald; CITO, Jurgen; LEITNER, Philipp. Continuous Experimentation: Challenges, Implementation Techniques, and Current Research. *Ieee Software*, [s.l.], v. 35, n. 2, p.26-31, mar. 2018.

[12] *Datanyze*. Disponível em https://www.datanyze.com/market-share/testing-and-optimization/Alexa%20top%20100K [acesso em 07/2019].

[13] *Built with*: https://trends.builtwith.com/analytics/a-b-testing/traffic/Top-100k [acesso em 07/2019].

[14] *SimilarTech*: https://www.similartech.com/categories/a-b-testing [acesso em 07/2019].

[15] *Google Optimize*: https://optimize.google.com/ [acesso em 07/2019].

[16] *Optimizely*: https://www.optimizely.com/ [acesso em 07/2019].

[17] *VWO*: https://vwo.com/ [acesso em 07/2019].

[18] *VentureHarbour*: https://www.ventureharbour.com/best-a-b-testing-tools/ [acesso em 07/2019].

[19] *HubSpot*: https://blog.hubspot.com/marketing/a-b-testing-tools [acesso em 07/2019].

[20] *Software Testing Help*: https://www.softwaretestinghelp.com/best-ab-testing-tools/ [acesso em 07/2019].

[21] *Quick Sprout*: https://www.quicksprout.com/ab-testing-tools/ [acesso em 07/2019].

[22] *FitSmallBusiness*: https://fitsmallbusiness.com/best-a-b-testing-tools/ [acesso em 07/2019].