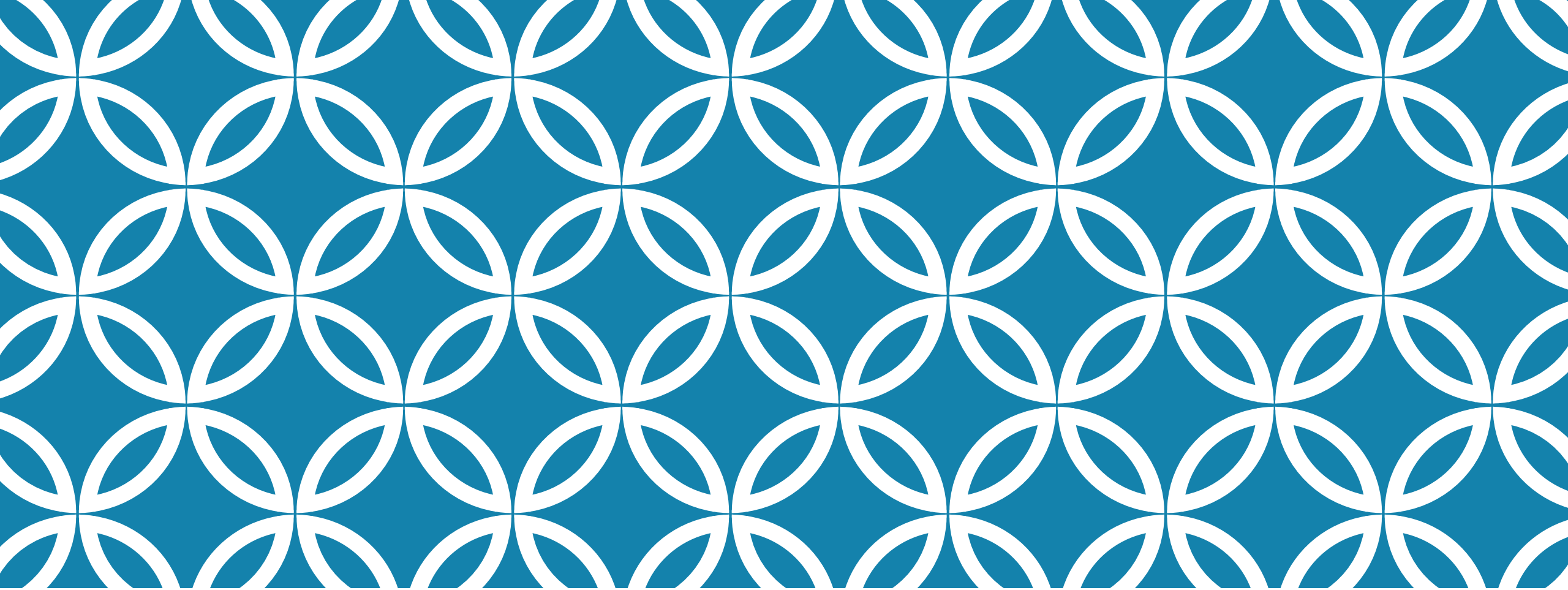

IMPACTOS DA ENTREGA CONTÍNUA EM PROJETOS DE SOFTWARE

Carlos Diego A. de Almeida
507580

AGENDA

1. Proposta de doutorado
2. Leis de Lehman
3. Medindo a qualidade de uma boa *CI CD*
4. Rails
5. TravisTorrent
6. GHtorrent (X)
7. pyGithub
8. DataSets

PROPOSTA DE DOUTORADO

Titulo: Um Estudo Empírico Multidimensional sobre os Desafios e Impactos da Entrega Contínua em Projetos de Software.

Um dos objetivos específicos:

1. Conduzir a mineração dos repositórios de software disponíveis para esse fim que possam trazer evidências sobre os impactos e desafios de *CI CD*

Lei	Descrição
Mudança contínua	Um programa usado em um ambiente do mundo real deve necessariamente mudar, ou se torna progressivamente menos útil nesse ambiente.
Aumento da complexidade	Como um programa em evolução muda, sua estrutura tende a tornar-se mais complexa. Recursos extras devem ser dedicados a preservar e simplicar a estrutura.
Evolução de programa de grande porte	A evolução de programa é um processo de autorregulação. Atributos de sistema como tamanho, tempo entre *releases* e número de erros relatados são aproximadamente invariáveis para cada *release* do sistema.
Estabilidade organizacional	Ao longo da vida de um programa, sua taxa de desenvolvimento é aproximadamente constante e independente dos recursos destinados ao desenvolvimento do sistema.
Conservação da familiaridade	Durante a vigência de um sistema, a mudança incremental em cada *release* é aproximadamente constante.
Crescimento contínuo	A funcionalidade oferecida pelos sistemas tem de aumentar continuamente para manter a satisfação do usuário.
Declínio de qualidade	A qualidade dos sistemas cairá, a menos que eles sejam modificados para refletir mudanças em seu ambiente operacional.
Sistema de *feedback*	Os processos de evolução incorporam sistemas de *feedback* multiagentes, *multiloop*, e você deve tratá-los como sistemas de *feedback* para alcançar significativa melhoria do produto.

MEDINDO A QUALIDADE DE UMA BOA *CI CD* FARLEY(2021)

Podemos determinar a qualidade da CI CD, medimos:

1. Taxa de falha de mudança – monitoramos com que frequência introduzimos um defeito nas diferentes partes do nosso processo
2. Tempo de recuperação de falha - medimos a quantidade de tempo quando nosso software não está em um estado liberável, ou seja: o tempo que leva para remediar o problema
3. Frequência - a frequência com que podemos liberar mudanças na produção,
4. Tempo de espera - quanto tempo leva para ir de um c*ommit* a um resultado liberável.

ESCOLHA DOS REPOSITÓRIOS DE DADOS E DO PROJETO A SER ESTUDADO

1. Quais repositórios possuíam informações sobre (gitlab, github actions, circleci)
   1. Integração contínua
   2. Controle de mudanças (issue tracker)
   3. Controle de versões (CVS)
2. Escolha do projeto
   1. Seja um projeto longevo (mais de 10 anos)
   2. Com dados disponíveis nas ferramentas de mineração
   3. E que tenham uma quantidade relevante de dados a serem cruzados

Com isso chegamos no projeto ***rails*** que tem boa parte do seu histórico no *TravisTorrent* e Suas issues no *Github*

TravisTorrent

Free and Open Travis Analytics for Everyone

TravisTorrent, a GHTorrent partner project, provides free and easy-to-use Travis CI build analyses to the masses through its open database.

TravisTorrent is the MSR'2017 Mining Challenge. By extracting cool facts from the data set, you can win awesome prizes, courtesy of Travis CI (deadline 20 Feb. 2017)!

TravisTorrent provides access to a database of hundreds of thousands of analyzed travis builds in **under 10 seconds**.

Access TravisTorrent now

FILTRAGEM

Haviam 47mil registros de builds de Jobs do rails de 2011 a 2016

Filtrei retirando todos os builds que não fossem na master

E retirei todas as tuplas dos Jobs mantendo apenas a tupla do build

Chegando a 15mil registros entre falhas e sucessos

Mostrar página do bigQuery

DADOS SOBRE AS ISSUES

1. Comecei utilizando da *API* do *Github* em *python*, mas por conta da limitação de consulta por usuário, resolvi tentar outra opção
2. Fui tentar utilizar do GHTorrent, parceiro do TravisTorrent, porem sem sucesso
3. Retornei ao uso do *github API* utilizando de vários usuário de do processo de paginação
4. Validei comparando os resultados com a busca da versão web

Mostrar jupyter notebook

TRABALHOS FUTUROS

1. Realizar uma análise e visualizações dos dados encontrados para as perguntas iniciais
2. Criar um modelo preditivo do crescimento do projeto e das suas características a partir do seu histórico