Java Generics Adoption

Daniella Angelos - Workshop Análise Estática

12 de Maio, 2015

Sumário

Introdução

Apresentação baseada no artigo *Java Generics Adoption: How New Features are Introduced, Championed, or Ignored*.

Estudo empírico da integração de Java Generics em softwares *open source*.

Hipóteses Estudadas

Afirmações analisadas

- **Hipótese 1:** Quando *Generics* é introduzido a um código-base, o número de *casts* de tipos é reduzido

- **Hipótese 2:** Introdução de classes genéricas definidas pelo usuário reduz duplicação de código

Questões de Pesquisa

Para melhor analisar a adoção de Generics, algumas perguntas foram feitas

- **Questão 01:** Generics será amplamente usado por membros de um projeto depois de introduzido a este projeto?
- **Questão 02:** Esforços em grande escala serão realizados para converter códigos antigos com uso de tipos normais à adoção de tipos genéricos?
- **Questão 03:** Suporte a Generics em IDEs influencia a adoção?

Projetos Estudados

Uma base de testes com os 20 mais populares projetos *open source* em Java foi gerada a partir do *ohloh.net*

Um foco maior é dado aos projetos *JEdit*, *Eclipse-CS* e *Squirrel-SQL*

Metodologia

Para analisar os 20 projetos, foi usado uma abordagem automatizada

1. Copiar os repositórios dos projetos a uma máquina local
2. Armazenar a versão de cada arquivo dos projetos em um arquivo intermediário
3. Analisar o uso de Generics em cada revisão e popular um banco MySQL
4. Analisar os dados do banco de diferentes formas, dependendo da informação que se deseja extrair

Caracterização de Dados

- Os projetos adotaram Generics?

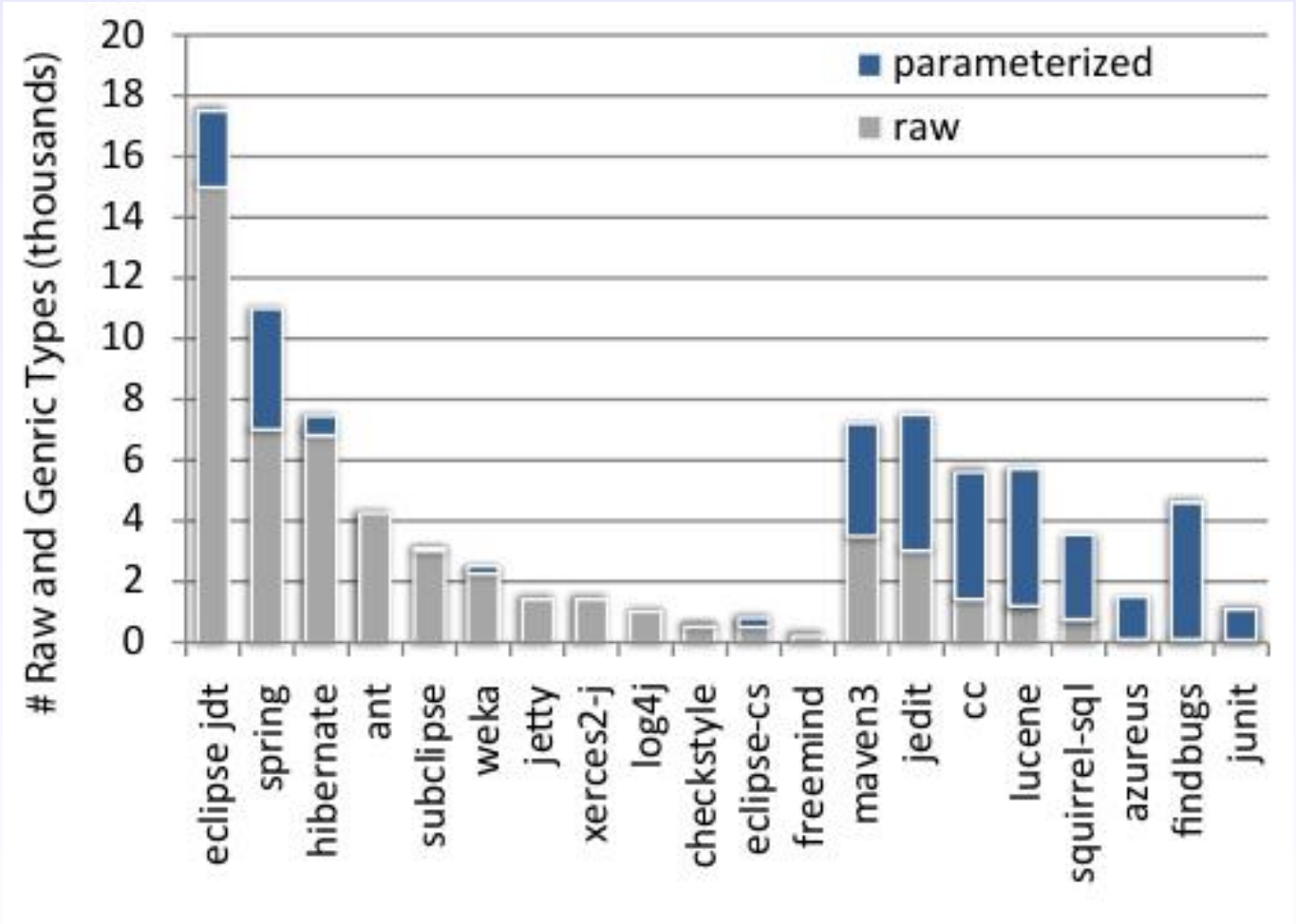

Figura: Tipos não-genéricos e parametrizados nos 20 projetos

Caracterização de Dados

Generics foi amplamente adotado pelos desenvolvedores?

- 532 desenvolvedores, 589.855 submissões
- 14% dos desenvolvedores criaram ou modificaram declarações genéricas
- 28% usaram tipos parametrizados
- 263 desenvolvedores submeteram código mais frequentemente
- Dentro deste grupo, 27% criaram ou modificaram declarações genéricas
- E 42% usaram tipos parametrizados

Caracterização de dados

- Quais os tipos parametrizados mais comuns?

Type	Parameterizations
List<String>	1066
ArrayList<String>	682
HashMap<String,String>	554
List<ObjectTreeNode>	376
List<ITableInfo>	322
Class<?>	314
List<TableColumnInfo>	304
Vector<String>	234
List<ArtifactStatus>	196
Collection<String[]>	166
List<Object[]>	132
Iterator<String>	124
ArrayList<MappedClassInfo>	114
Set<String>	102

Figura: Número de parametrizações de diferentes tipos genéricos in Squirrel-SQL

Investigação das Hipóteses

- **Generics reduz o uso de casts**

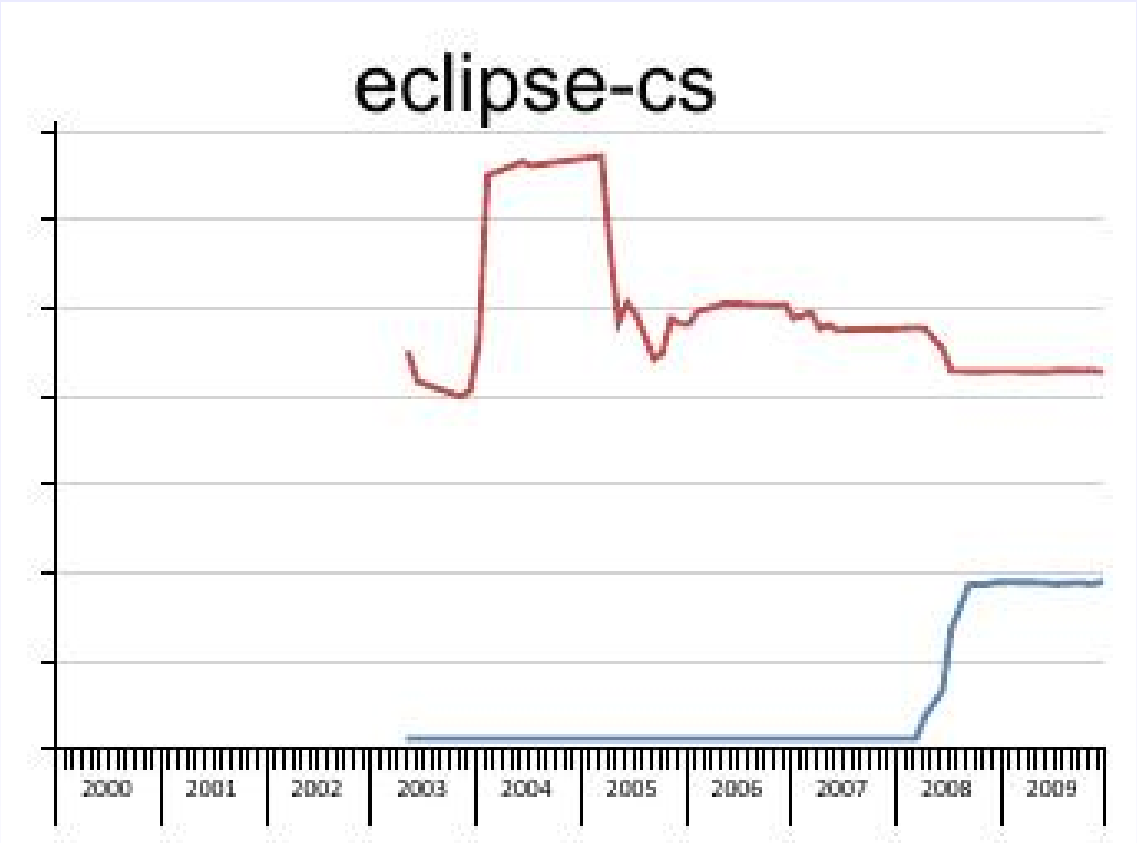

Figura: Número de casts (linha vermelha) vs número de parametrizações (linha azul)

Investigação das Hipóteses

- **Generics reduz o uso de casts**

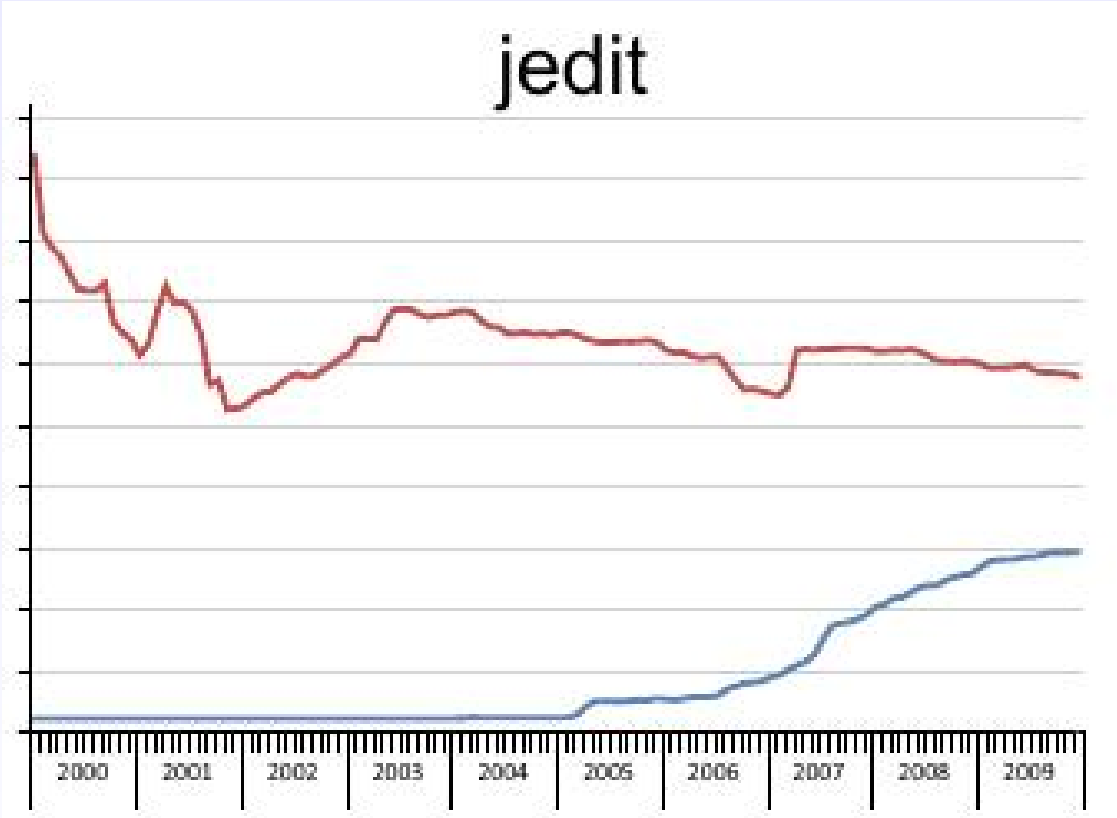

Figura: Número de casts (linha vermelha) vs número de parametrizações (linha azul)

Investigação das Hipóteses

- **Generics reduz o uso de casts**

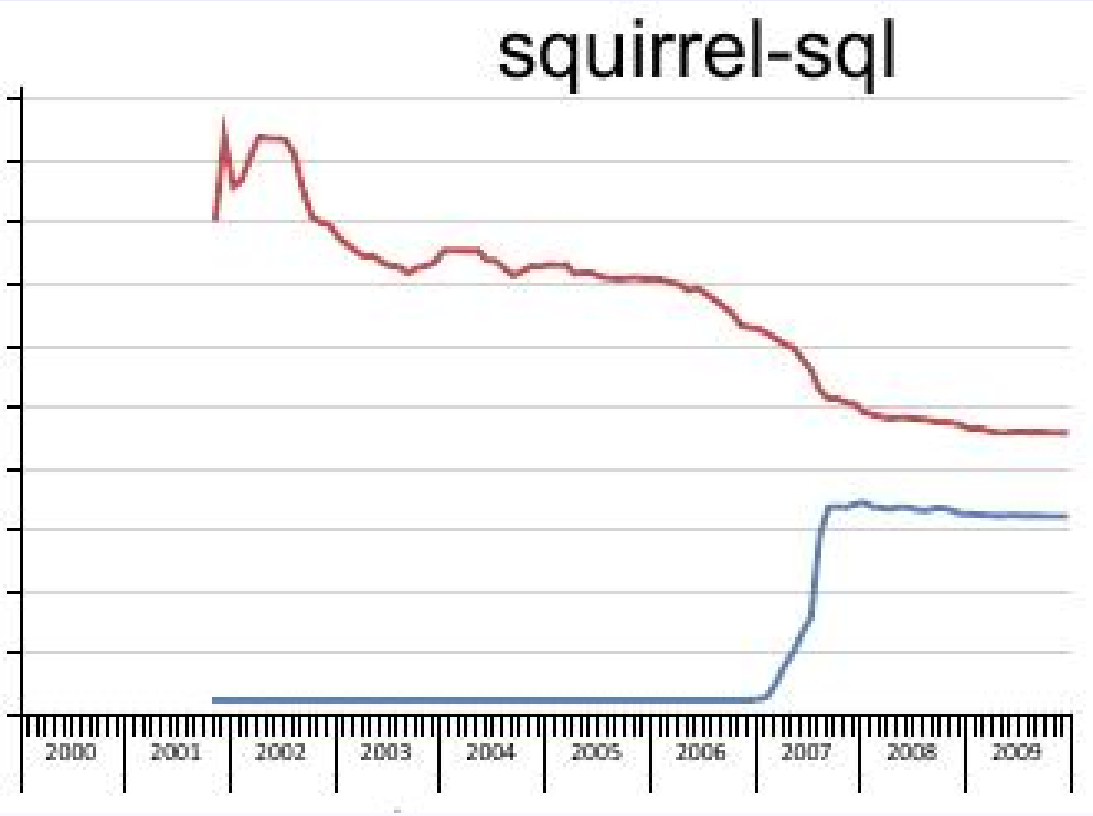

Figura: Número de casts (linha vermelha) vs número de parametrizações (linha azul)

Investigação das Hipóteses

Os gráficos não sugerem uma forte relação entre a diminuição do uso de casts e o uso de tipos parametrizados

- O número de casts flutua significativamente na fase inicial de todos os 3 projetos
- A queda do uso de casts aparece mesmo antes de generics começar a ser adotado

Investigação das Hipóteses

Entretanto, os gráficos sugerem que podem ter existido situações em que o uso de Generics reduziu o número de casts.

Mas os dados coletados não dão suporte à **Hipótese 1**.

Investigação das Hipóteses

- **Generics reduz duplicação de código**

O total de linhas que seriam duplicadas é calculado usando o número de parâmetros únicos (P), o número de linhas de códico (LOC) aplicados à fórmula:

$$D = LOC * (P - 1)$$

Que estima a quantidade de código adicional que implementações de código não-genérico deveria prover para cada parâmetro P.

Investigação das Hipóteses

- **Generics reduz duplicação de código**

Para essa verificação, foram analisados 27 tipos parametrizados definidos por usuários para estimar o impacto na duplicação de código

As classes genéricas tinham um total de 365 variações de parâmetros

O tamanho médio dos códigos era de 378 linhas, e os tipos foram alterados um total de 775 vezes (média: 28)

Investigação das Hipóteses

Com esses 27 tipos genéricos, foram estimados uma prevenção de 107.985 linhas de código

Portanto, isso dá suporte a **Hipótese 2**, mas o impacto não é tão grande quanto o esperado.

Adoção de Generics

- **Questão 1**

Para responder a essa pergunta, foram analisados tipos parametrizados adicionados e removidos ao longo do tempo

O padrão mais observado foi a maior parte da manipulação de tipos genéricos sendo feita por um ou dois usuários na maioria dos projetos

Esse comportamento indica que não há adoção ampla dos usuários do projeto ao uso de Generics, uma vez introduzido

Adoção de Generics

- **Questão 2**

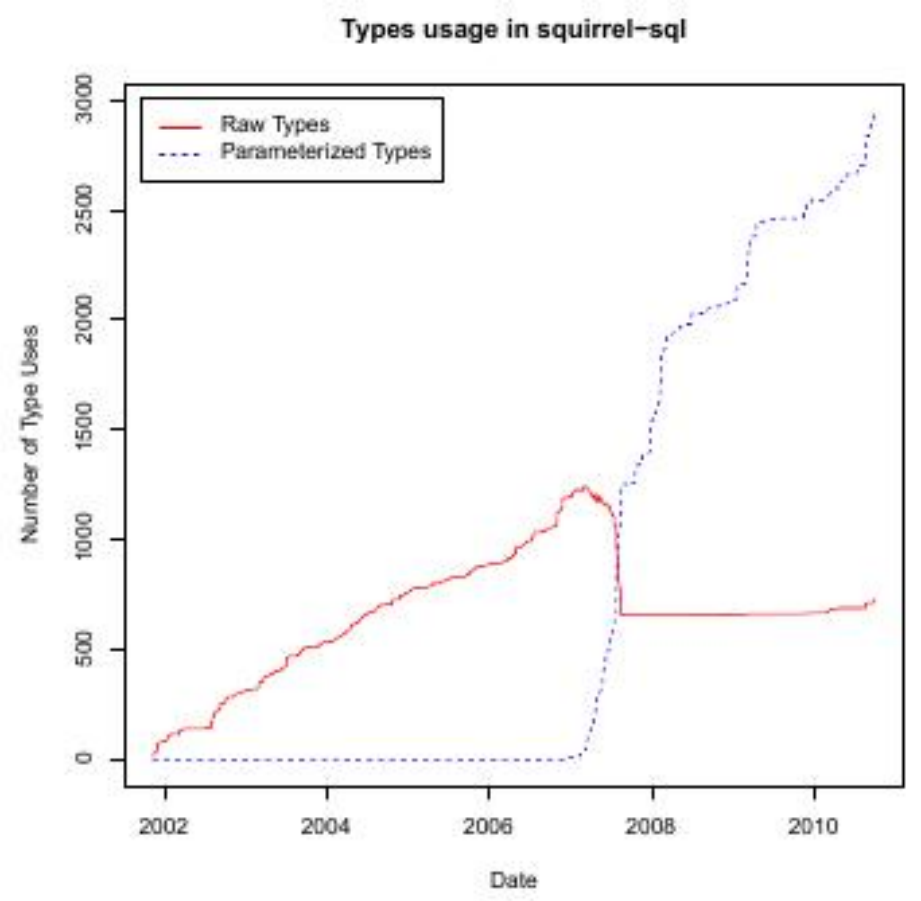

Figura: Comparação entre o aumento do número de tipos não-genéricos (linha vermelha) e o de tipos parametrizados (linha azul)

- **Questão 2**

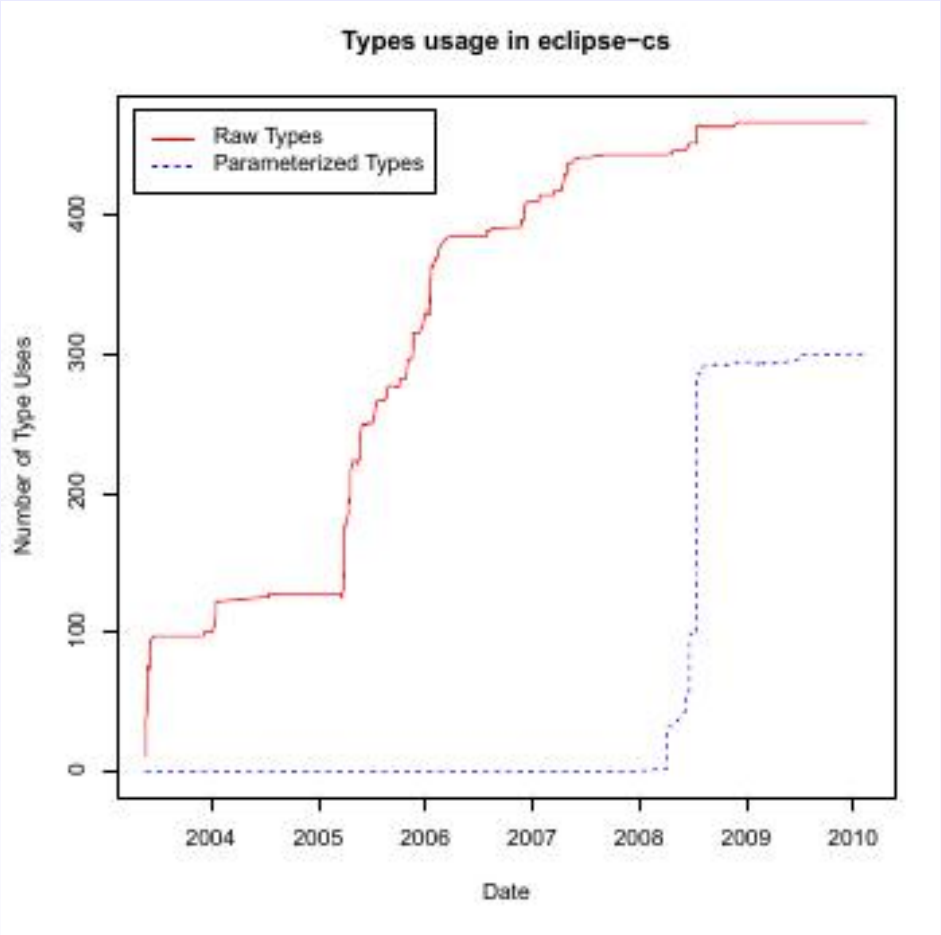

Figura: Comparação entre o aumento do número de tipos não-genéricos (linha vermelha) e o de tipos parametrizados (linha azul)

- **Questão 2**

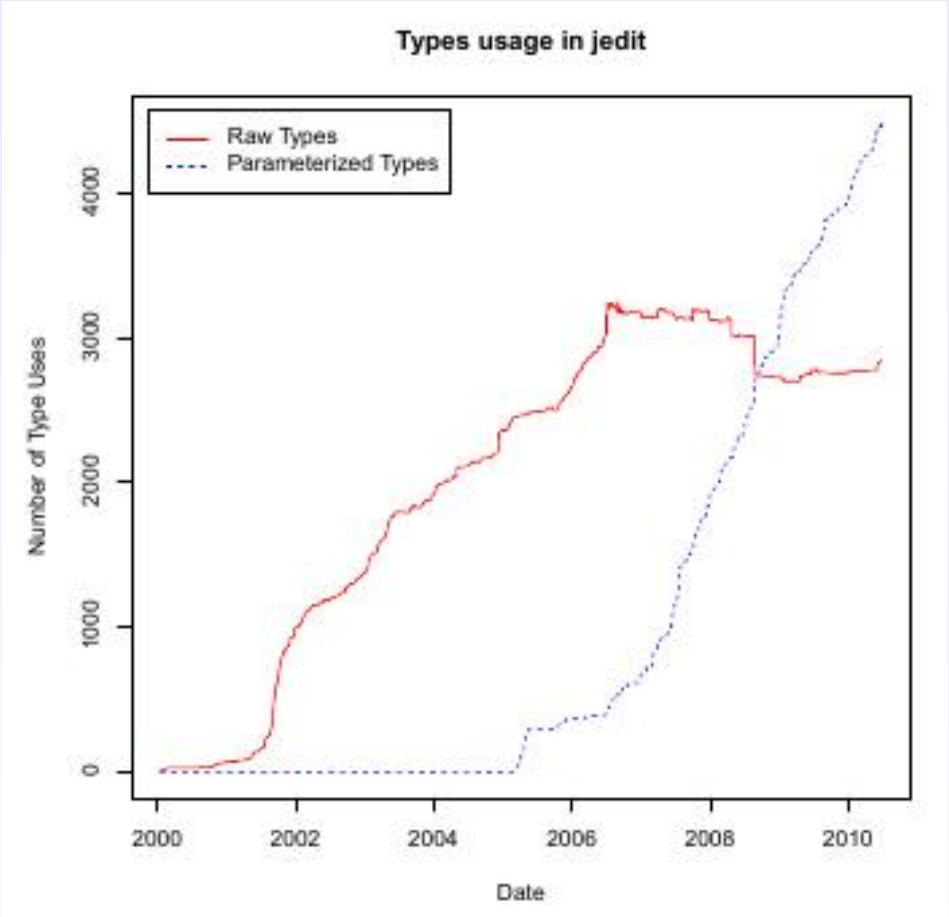

Figura: Comparação entre o aumento do número de tipos não-genéricos (linha vermelha) e o de tipos parametrizados (linha azul)

Adoção de Generics

Apesar de vermos alguns esforços de migrações, **a maioria dos projetos não mostram uma conversão em larga escala de tipos não-genéricos a tipos parametrizados**.

Adoção de Generics

- **Questão 3**

Generics foi lançado na versão 5.0 de Java, em Setembro de 2004, entretanto, Eclipse não dava suporte até o lançamento da versão 3.1, em Junho de 2005.

O desafio então, foi analisar o aumento de adoções de Generics a partir do lançamento da versão do Eclipse com suporte para tal

Já que não houveram grandes variações a partir deste período, os dados coletados indicaram que **ausência de suporte a Generics por IDE não tem impacto sobre sua adoção**.

Fim da Apresentação

- Obrigada pela atenção