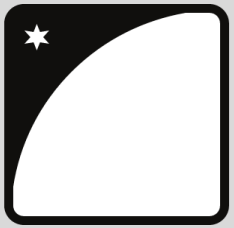


Caelum
Ensino e Inovação

Does the Act of Refactoring Really Make Code Simpler? A Preliminary Study

Francisco Zigmund Sokol, Mauricio Finavaro Aniche, Marco Aurélio Gerosa

francisco.sokol@usp.br, {aniche, gerosa}@ime.usp.br

Refatoração

Pequenas mudanças no código para aprimorar a qualidade interna sem alterar as funcionalidades de um software.

- Legibilidade?
- Coesão?
- Acoplamento?
- **Complexidade?**
  - **Complexidade Ciclomática.**

Exemplo - Complexidade Ciclomática

```
public void Employee {
    //...
    public void pay(double value) {
        double factor = 1.0;
        if (isManager()) {
            factor = 0.9;
        }
        this.money += value * factor;
    }
    public void payOvertime(int hours) {
        double factor = 1.0;
        if (isManager()) {
            factor = 0.9;
        }
        this.money += hours * getHourValue()
            * factor;
    }
}
```

CC = 4

Exemplo - Refatoração

```
public void Employee {
    //...
    public void pay(double value) {
        double factor = 1.0;
        if (isManager()) {
            factor = 0.9;
        }
        this.money += value * factor;
    }
    public void payBonus(double value) {
        double factor = 1.0;
        if (isManager()) {
            factor = 0.9;
        }
        this.bonus += value * factor;
    }
    public void payOvertime(int hours) {
        double factor = 1.0;
        if (isManager()) {
            factor = 0.9;
        }
        this.money += hours * getHourValue()
            * factor;
    }
}
```

Exemplo - Refatoração

```
public void Employee {
    //...
    public void pay(double value) {
        double factor = 1.0;
        if (isManager()) {
            factor = 0.9;
        }
        this.money += value * factor;
    }
    public void payBonus(double value) {
        double factor = 1.0;
        if (isManager()) {
            factor = 0.9;
        }
        this.bonus += value * factor;
    }
    public void payOvertime(int hours) {
        double factor = 1.0;
        if (isManager()) {
            factor = 0.9;
        }
        this.money += hours * getHourValue()
            * factor;
    }
}
```

Extract method

```
public void Employee {
    //...
    public void pay(double value) {
        this.money += value * calculateFactor();
    }
    public void payBonus(double value) {
        this.bonus += value * calculateFactor();
    }
    public void payOvertime(int hours) {
        this.money += hours * getHourValue()
           * calculateFactor();
    }
    private double calculateFactor() {
        double factor = 1.0;
        if (isManager()) {
            factor = 0.9;
        }
        return factor;
    }
}
```

Exemplo - Refatoração

```
public void Employee {
    //...
    public void pay(double value) {
        double factor = 1.0;
        if (isManager()) {
            factor = 0.9;
        }
        this.money += value * factor;
    }
    public void payBonus(double value) {
        double factor = 1.0;
        if (isManager()) {
            factor = 0.9;
        }
        this.bonus += value * factor;
    }
    public void payOvertime(int hours) {
        double factor = 1.0;
        if (isManager()) {
            factor = 0.9;
        }
        this.money += hours * getHourValue()
            * factor;
    }
}
```

CC = 6

Extract method

```
public void Employee {
    //...
    public void pay(double value) {
        this.money += value * calculateFactor();
    }
    public void payBonus(double value) {
        this.bonus += value * calculateFactor();
    }
    public void payOvertime(int hours) {
        this.money += hours * getHourValue()
            * calculateFactor();
    }
    private double calculateFactor() {
        double factor = 1.0;
        if (isManager()) {
            factor = 0.9;
        }
        return factor;
    }
}
```

CC = 5

Objetivo

A complexidade ciclomática reduz ao aplicar uma refatoração?

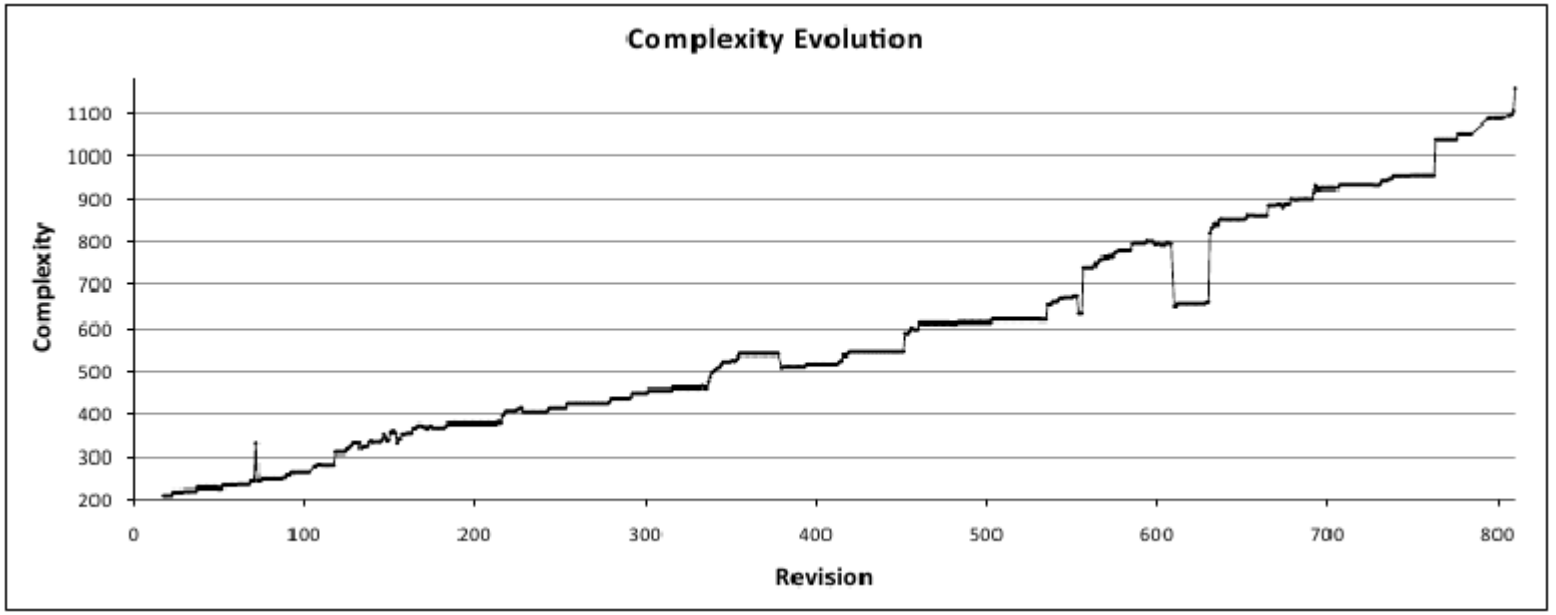
Replicação do trabalho de Soetens e Demeyer (2010).

Trabalhos relacionados

M. Alshayeb (2009)	Adaptabilidade, manutenabilidade, legibilidade e testabilidade.	Três projetos OS.	**Não apresenta melhoria significativa.**
Du Bois et al (2006)	Legibilidade.	Experimento controlado.	**Apresenta melhoria.**
Stroggylos e Spinellis (2007)	Complexidade, coesão, acoplamento.	Três projetos OS.	**Apresenta piora.**
Geppert et al (2005)	Densidade de bugs e capacidade de mudança.	Projeto da indústria.	**Apresenta melhoria.**

Soetens e Demeyer

Análise da evolução da Complexidade Ciclomática de ~800 versões do projeto de código aberto PMD:



Soetens e Demeyer

	Decrease CC	Equalized CC	Increased CC
Documented Refactoring	14	7	12
No Refactoring Documented	27	580	136

- Poucas refatorações removiam código duplicado.
- Commits de refatoração junto com novas funcionalidades.
- Mudanças simples como mover métodos e renomear variáveis.

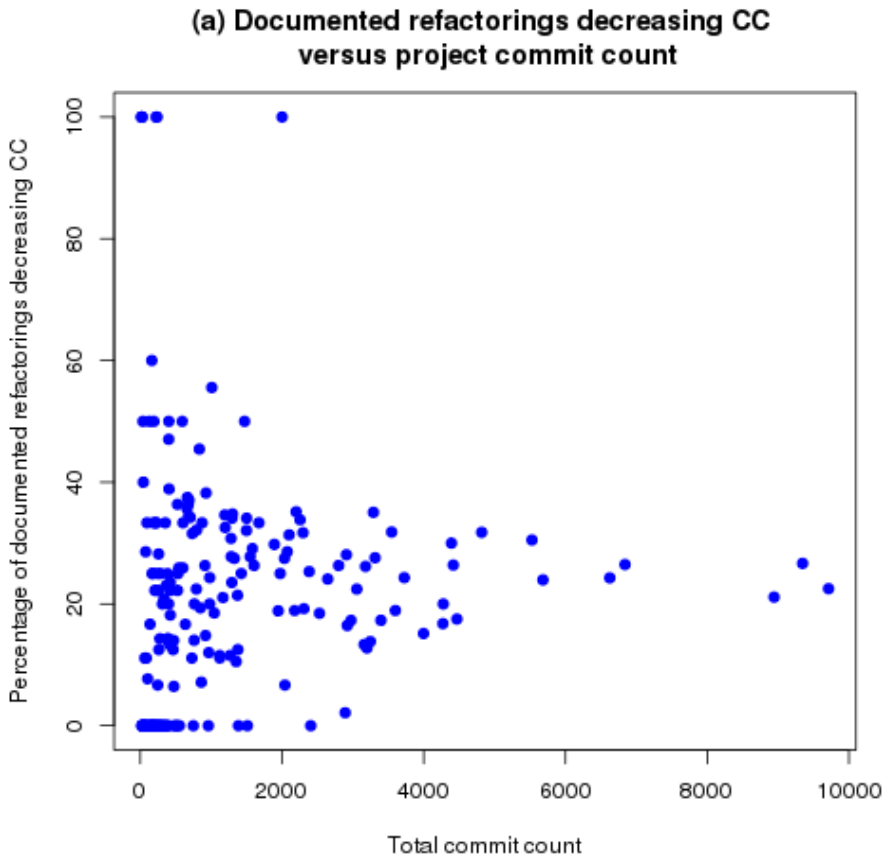
Protocolo de replicação

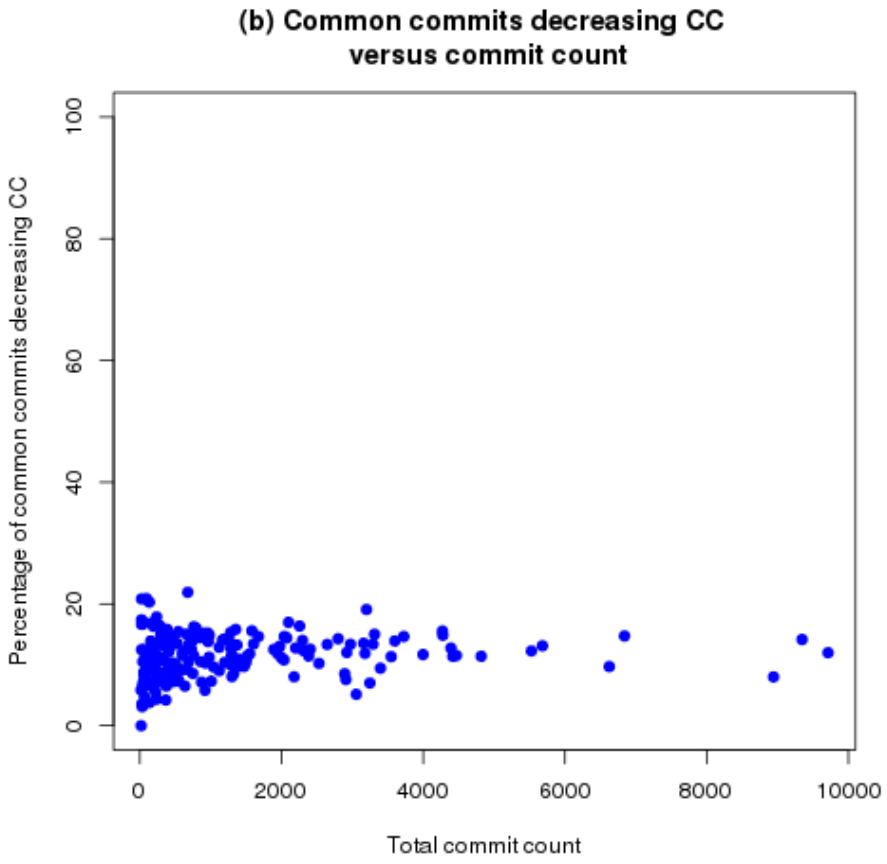
- 256 projetos Java da Apache.
- Aproximadamente 500 mil commits.
- Classificação de uma refatoração por meio da mensagem de commit.
- Complexidade Ciclomática pré-calculada no MetricMiner (http://metricminer.org.br/).

Resultados

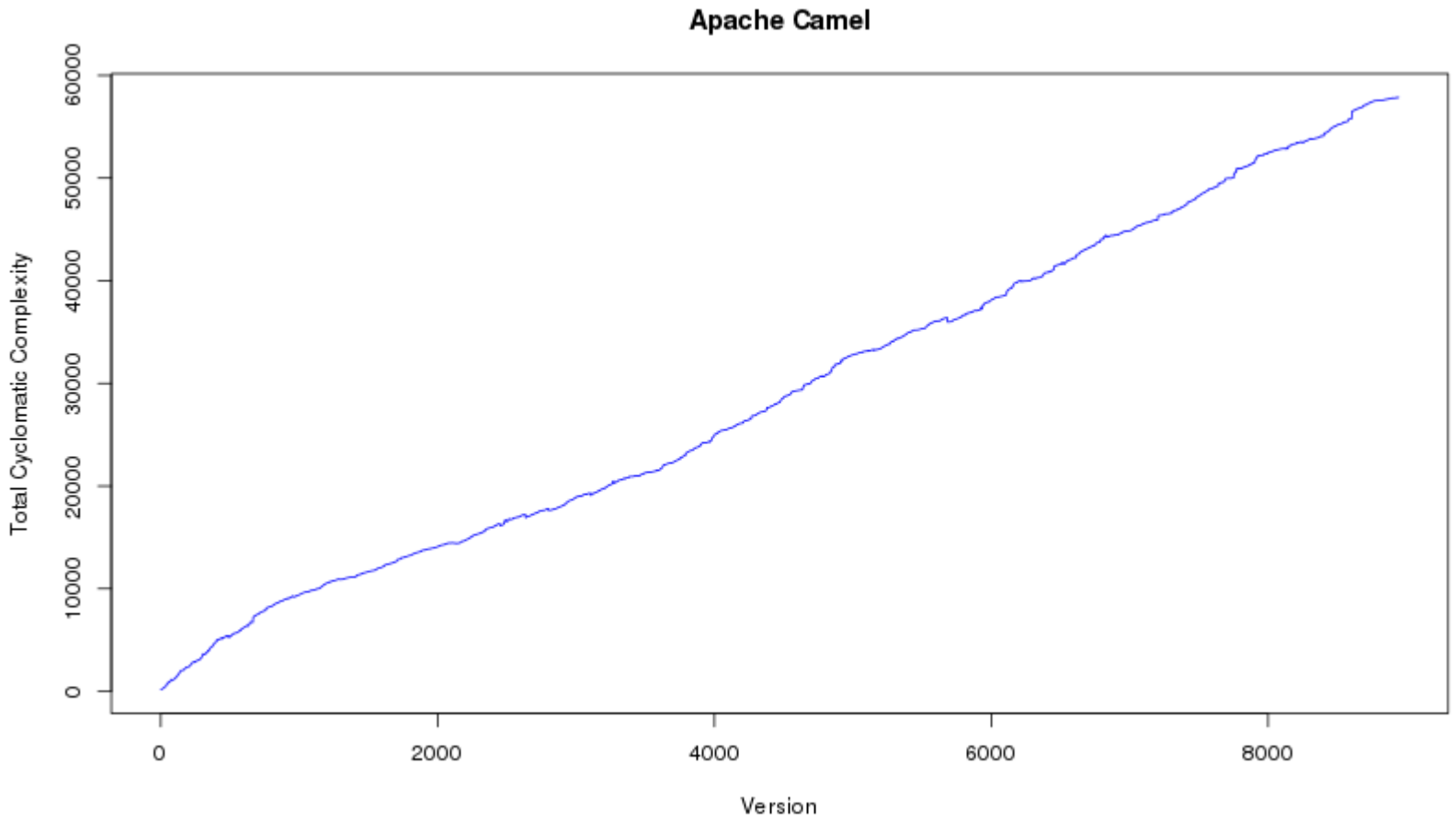
	Decrease CC	Equalized CC	Increased CC
Documented Refactoring	1504	1603	3230
No Refactoring Documented	30145	99580	121239

Resultados

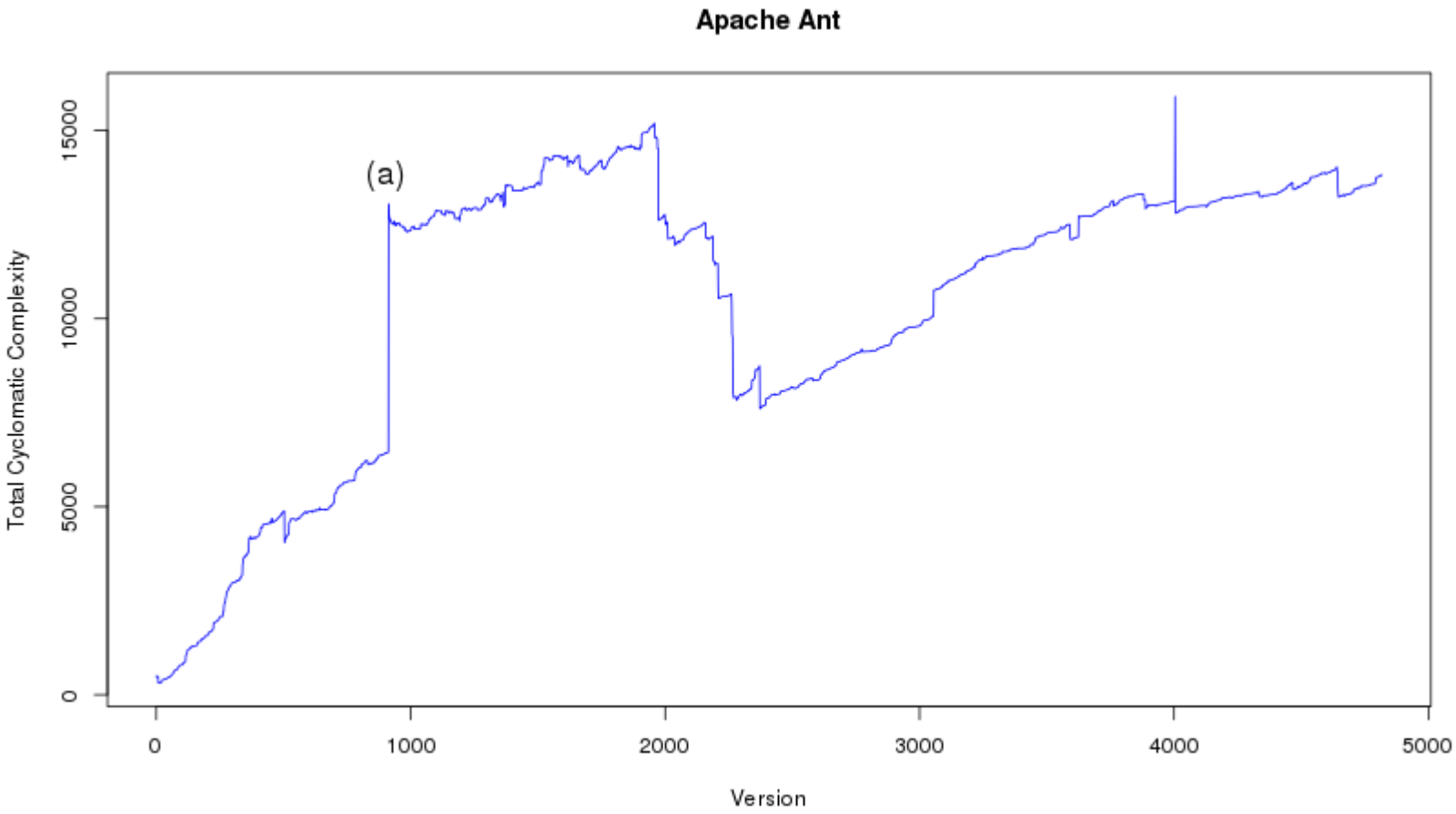

(a) Documented refactorings decreasing CC versus project commit count
Percentage of documented refactorings decreasing CC
0
20
40
60
80
100
0
2000
4000
6000
8000
10000
Total commit count


(b) Common commits decreasing CC versus commit count
Percentage of common commits decreasing CC
0
20
40
60
80
100
0
2000
4000
6000
8000
10000
Total commit count

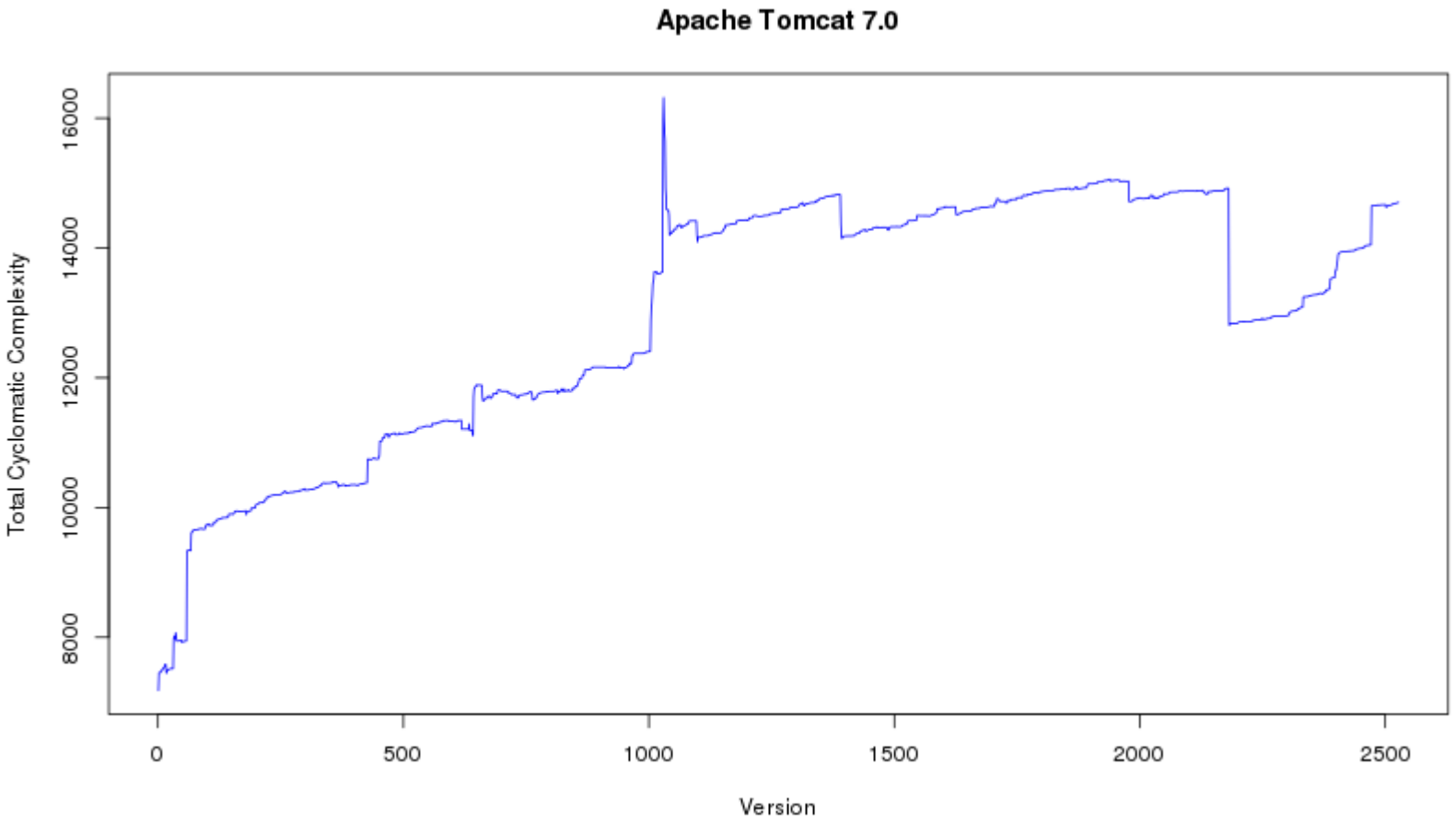
Resultados


Apache Camel
Total Cyclomatic Complexity
0
10000
20000
30000
40000
50000
60000
0
2000
4000
6000
8000
Version

Resultados


Apache Ant
(a)
Total Cyclomatic Complexity
0
5000
10000
15000
0
1000
2000
3000
4000
5000
Version

Resultados


Apache Tomcat 7.0
Total Cyclomatic Complexity
8000
10000
12000
14000
16000
0
500
1000
1500
2000
2500
Version

Análise qualitativa

Category	Quantity (Percentage)
Code Removed	17 (68%)
Extract Class	5 (20%)
Extract Method	1 (4%)
Class Removed	6 (24%)
Extract Interface	1 (4%)

Table 3. Distribution of the 25 Commits that Decreased CC.

Category	Quantity (Percentage)
Functionality added	15 (60%)
Move Method	1 (4%)
Rename Method	2 (8%)
Extract Super Class	4 (16%)
Extract Class	7 (28%)
Pull Up Method	3 (12%)

Table 4. Distribution of the 25 Commits that Increased CC.

Análise qualitativa

- Muitos commits de refatoração adicionavam também novas funcionalidades - *floss refactoring* (Murphy-Hill et al 2010).
- Commits grandes.
- Melhorias principalmente em legibilidade.

Ameaças à validade

- O método de classificação de refatorações é simples.
- Poucos dados utilizados na análise qualitativa.
- Complexidade Ciclomática total da classe pode não ser a métrica ideal para medir complexidade.
- Classes de teste não foram ignoradas.
- Conjunto de dados limitado aos projetos da Apache.

Conclusão e trabalhos futuros

- Não é possível afirmar que a prática de refatoração de código diminui a Complexidade Ciclomática.

- A análise qualitativa indica que a maioria das refatorações traz melhorias a legibilidade.

Conclusão e trabalhos futuros

- Utilização de outras métricas de para medir outros aspectos de qualidade de código.

- Os padrões diferentes de evolução da Complexidade Ciclomática de cada projeto podem ser investigados.

Obrigado!

Perguntas?