Um Modelo de Predição de Amplitude da Propagação de Modificações Contratuais em Software Orientado por Objetos

Kecia Aline Marques Ferreira

Orientadora: Prof.ª Mariza Bigonha
Co-orientador: Prof. Roberto Bigonha
Colaborador: Prof. Bernardo de Lima

Um Modelo de Predição de Amplitude da Propagação de Modificações Contratuais em Software Orientado por Objetos

- Contextualização
- Problema
- Objetivos
- Solução Proposta
  - Valores referência para métricas de software OO
  - Métrica de Coesão de Responsabilidade
  - Um estudo sobre a evolução da estrutura de software OO
  - K3B - Definição e avaliação do modelo proposto
- Conclusões

Contextualização

- Mais de 70% do custo total de um software corresponde à atividade de manutenção

  Meyer (95), Pfleeger (98)

- A medição, a avaliação e o controle da **manutenibilidade** do software são importantes durante as fases de desenvolvimento e operação do software para a predição de custos e riscos

- **Modificabilidade**, a facilidade de se realizar modificações em um software, é uma característica importante da manutenibilidade

- **Propagação de modificações** refere-se ao processo em que uma modificação em determinado elemento ou grupo de elementos de um software ocasiona modificações em outros elementos sucessivamente

Contextualização

- O controle de **propagação de modificações** é de grande relevância na gestão de modificações em software

- Alguns trabalhos têm sido realizados com o objetivo de criar recursos para avaliação e estimar propagação de modificações

  Myers (1975), Chaumun et al. (1999), Mirarab et al. (2007) e Li et al. (2010)

- A maior parte desses modelos propostos:

  – Avalia como a modificação de um determinado módulo será propagada pelo software

    - Mas um processo de modificação pode ser iniciado por modificações em mais de um módulo

Contextualização

- Conta o número de módulos que serão afetadas em um processo de modificação
  - Essa é uma informação importante, mas não fornece um indicador completo da dificuldade de se realizar modificação no software
  - Um módulo pode ser afetado mais de uma vez em um processo de modificação
- Tem implementação complexa
  - Isso pode inibir a aplicação do modelo na prática
- Possui definição que não viabiliza a identificação dos fatores que possam ser responsáveis pela alta propagação de modificações

Problema

- Não há, ainda, um modelo que avalie com fidelidade a propagação de modificações em software orientado por objetos
- Diante de uma alta propagação de modificações, é necessário:
  - Identificar os fatores que possam ser a causa dessa alta propagação
    - Para isso, é necessário avaliar o software, idealmente por meio de métricas
  - Decidir sobre a reestruturação do software, para alcançar melhor modificabilidade

Problema

- Há uma grande variedade de métricas de software OO, mas seus valores referência ainda não são conhecidos

  Tempero (2008)

- Para alguns fatores, como coesão interna de módulos, não há um consenso sobre uma métrica para sua avaliação

  Counsell et al. (2004), Mäkelä & Leppanen (2007)

Objetivos

- Definição de um modelo de predição de amplitude da propagação de modificações contratuais em software orientado por objetos
  - **Contrato de uma classe**
    - Serviços exportados pela classe
    - Pré-condições: deve ser satisfeita para que a classe garanta a realização do serviço
    - Pós-condições: condição resultante do serviço
  - **Modificações contratuais**: aquelas que provocam alteração do contrato da classe

Objetivos

- O modelo, denominado K3B, visa responder a seguinte questão:

Qual é o número médio de passos de modificações, *E(n,i),* para que um software com *n* módulos alcance estabilidade se ele teve *i* de seus módulos modificados?

Passo de modificação: conjunto de atividades de modificações em um módulo desde o início de uma dada modificação até o momento em que ele é declarado como atualizado.

Objetivos

- Identificação de valores referência para métricas de software OO, em especial para aquelas utilizadas com o modelo proposto
- Definição de uma métrica para avaliar coesão interna de classes (Coesão de Responsabilidade)
- Estudo da evolução da estrutura de software orientado por objetos

Valores Referência de Métricas de Software Orientado por Objetos

Motivação

- Métricas de software
  - São essenciais para medir, avaliar, controlar e melhorar produtos e processos de software
  - Não têm sido utilizadas de forma ampla na indústria ainda
- Motivo possível: para a maior parte das métricas, os **valores referência** ainda não são conhecidos
- Objetivo
  - Identificar valores referência para um conjunto de métricas de software OO, em particular aquelas que poderão ser utilizadas em associação a K3B

Metodologia

- Metodologia
  - Foram utilizados 40 softwares abertos de 11 domínios de aplicações diferentes (sourceforge.net) desenvolvidos em Java
  - O estudo envolve 26.202 classes
  - Critério de seleção de softwares para análise:
    - Tamanhos diversificados
    - Domínios de aplicação diversos
    - *Bytecode* disponível ou facilidade de sua geração

Metodologia

- Métricas avaliadas:
  - **COF** (fator acoplamento)
    - a / (n*(n-1))
    - ***a*** é o número de conexões existentes no software
    - ***n*** é o número de classes do software
  - **LCOM** (*lack of cohesion in method*s): diferença entre o número de pares de métodos sem similaridade e o número de pares de métodos com similaridade
  - **DIT** (*depth of inheritance tree*): distância máxima da classe à raiz da sua árvore de herança
  - **Conexões aferentes**: número de classes que dependem da classe
  - **Número de atributos públicos**
  - **Número de métodos públicos**

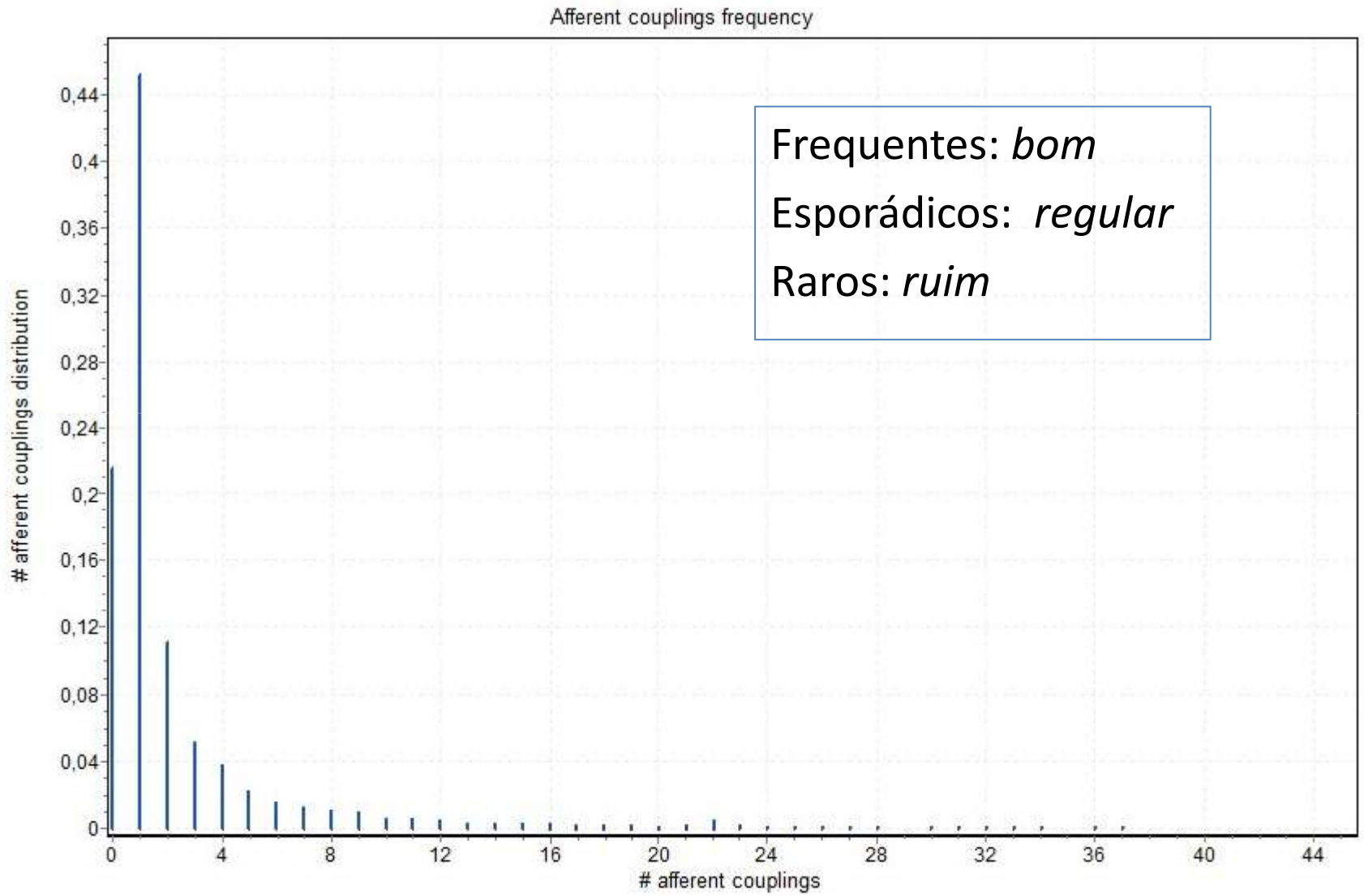
Metodologia

- Foram identificados valores referência gerais, por tamanho de software (em número de classe), por domínio de aplicação e por tipo (*framework*, biblioteca e ferramenta).

- Ajuste dos dados: as medidas coletadas foram ajustadas a diversas distribuições de probabilidades

- Análise dos ajustes e identificação de valores referência para as métricas

Metodologia



Frequentes: *bom*

Esporádicos: *regular*

Raros: *ruim*

Valores Referência Identificados

Fator	Nível	Métrica	Valor Referência
Conectividade	Sistema	COF	Bom: até 0,02 Regular: entre 0,02 e 0,14 Ruim: ≥ 0,14
	Classe	#conexões aferentes	Bom: 1 Regular: de 1 a 20 Ruim: > 20
Ocultação de Informação	Classe	#atributos públicos	Bom: 0 Regular: de 1 a 10 Ruim: > 10
Tamanho da Interface	Classe	#métodos públicos	Bom: de 0 a 10 Regular: de 11 a 40 Ruim: > 40
Herança	Classe	DIT	Valor típico: 2
Coesão interna	Classe	LCOM	Bom: 0 Regular: de 1 a 20 Ruim: > 20

Avaliação dos Valores Referência

- Para avaliar o valores referência identificados, foram realizados 2 estudos de caso

- Estudo de Caso 1:

  - **Objetivo**: avaliar a capacidade dos valores referência em identificar classes com deficiências do ponto de vista de sua estrutura

  - Foram utilizadas todas as classes do conjunto de softwares utilizado para derivar os valores referência

  - O estudo de caso consistiu em inspecionar as classes com métricas na faixa *ruim*

Avaliação dos Valores Referência

- Foram selecionadas as classes com
  - Conexões aferentes > 20 e LCOM > 20 e #atributos públicos > 10: 24 classes
- Os resultados dessa avaliação indicam que a aplicação desses valores referência auxiliaram adequadamente na identificação de classes com deficiências estruturais

- Estudo de Caso 2:
  - **Objetivo**: verificar se os valores referência podem ser utilizados para identificar corretamente classes de boa estrutura
  - Para isso, foi utilizado um software cuja estrutura tem sido qualitativamente avaliada como boa: JHotDraw

Avaliação dos Valores Referência

- Resultados:
  - **COF**: 0,003, dentro da faixa *bom*
  - **Conexões aferentes:** 97% das classes do software têm qualidade boa ou pelo menos regular
  - **LCOM:** 80% das classes têm coesão boa ou pelo menos regular
  - **DIT**: 70% das classes, possuem DIT até 2; DIT médio do software é 2.
  - **# Atributos públicos**: praticamente a totalidade das classes do software são avaliadas como boas ou pelo menos regulares
  - **# Métodos públicos:** 99% das classes do software são avaliadas como boas ou pelo menos regulares.

Métrica Coesão de Responsabilidade (COR)

Métrica Coesão de Responsabilidade

- Para benefício da modularidade, é importante a criação de classes com alta coesão
- Há várias métricas de coesão de classes propostas
- Não há um consenso sobre a forma mais adequada de se medir coesão
- Problemas:
  - Algumas métricas avaliam ausência de coesão
  - Definição complexa
  - Interpretação difícil dos valores

Métrica Coesão de Responsabilidade

- Objetivo
  - Identificar uma métrica
    - Adequada ao uso com K3B
      - Valores baixos representam baixa coesão, valores altos representam boa coesão
      - Valores no intervalo [0,1], preferencialmente
    - Com valores de interpretação mais simples

Métrica Coesão de Responsabilidade

- Ideia da métrica Coesão de Responsabilidade (COR): indicar o número de responsabilidades que um classe implementa

- Conceitos:

  – Responsabilidade: conjunto de métodos com relacionamento entre si

  – Relacionamento: dois métodos de uma classe *C* estão relacionados se usam pelo menos um atributo de *C* em comum, ou se um utiliza o outro, ou se utilizam direta ou indiretamente métodos em comum

  – Propriedade transitiva de relacionamento: se *a* está relacionado com *b* e *b* está relacionado com *c*, então *a* esta relacionado com *c*.

Métrica Coesão de Responsabilidade

- Definição da métrica:
  - **C**: conjunto dos conjuntos disjuntos de métodos com relacionamentos entre si
  - **r = |C|** : número de responsabilidades
  - COR = 1/r, se *r*>0
    COR = 0, caso contrário
- Exemplo:
  - Uma classe com COR igual a 0,5 possui 2 responsabilidades
  - Uma classe com COR igual a 1 possui uma responsabilidade

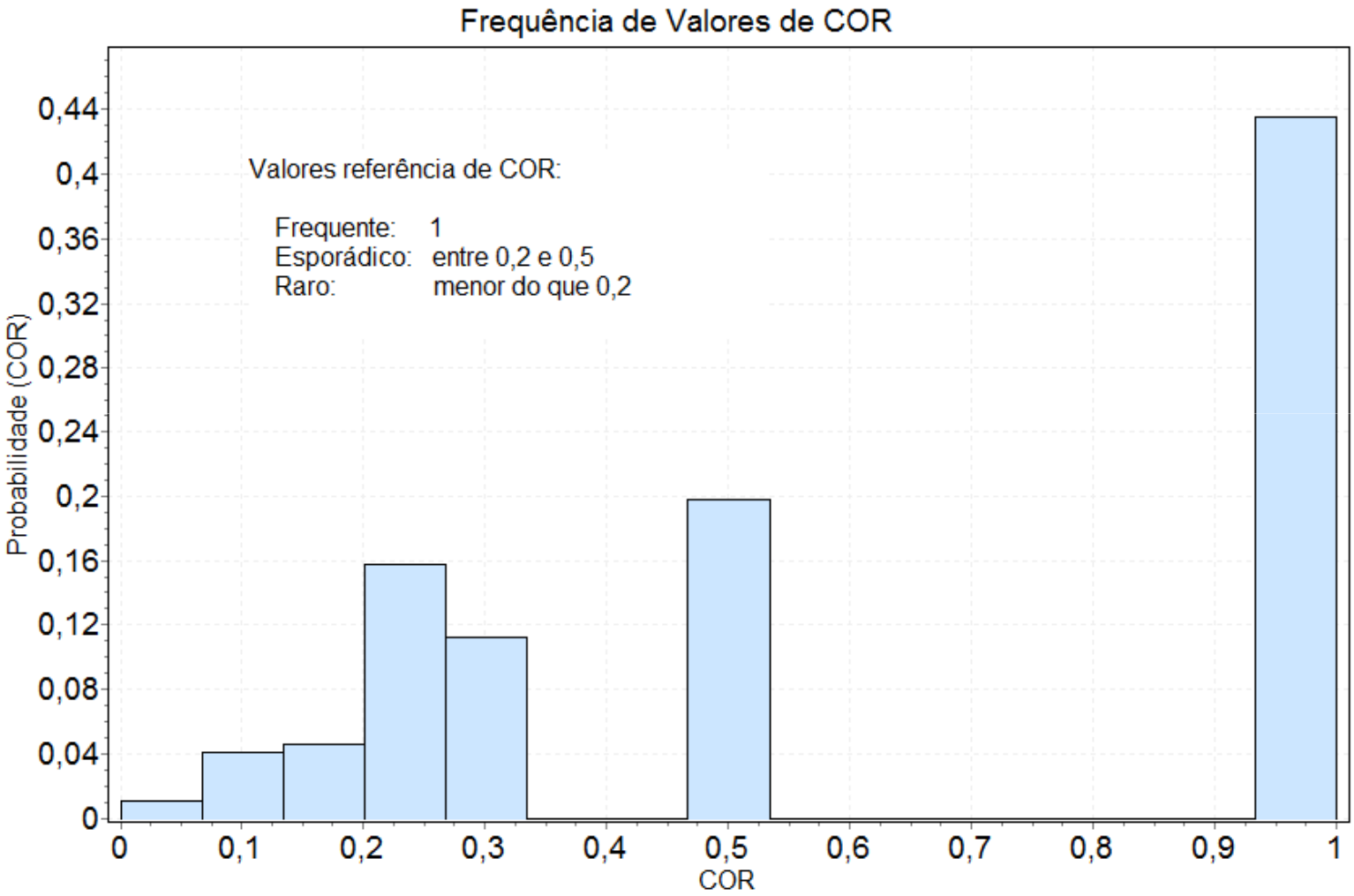
Métrica Coesão de Responsabilidade

- Avaliação de COR
  - Seus resultados foram comparados aos de LCOM em 14 classes
    - Em todos os casos avaliados, o resultado de COR corresponde ao número de responsabilidades observados
  - COR foi utilizada para auxiliar a identificação de necessidades de reestruturações em um software com cerca de 60 classes
  - Das 60 classes avaliadas, COR não capturou falhas de coesão em 4 classes, no seguinte caso:
    - Classe com apenas 1 método que executa todos os serviços, tais como interface com o usuário e persistência de dados

Métrica Coesão de Responsabilidade

- Considerações
  - A base conceitual de COR (avaliar relacionamento entre métodos da classe) é a mesma de outras métricas como LCOM
  - Porém, a interpretação dos valores gerados por COR é mais simples
  - COR foi utilizada na implementação de K3B

COR – Valores Referência

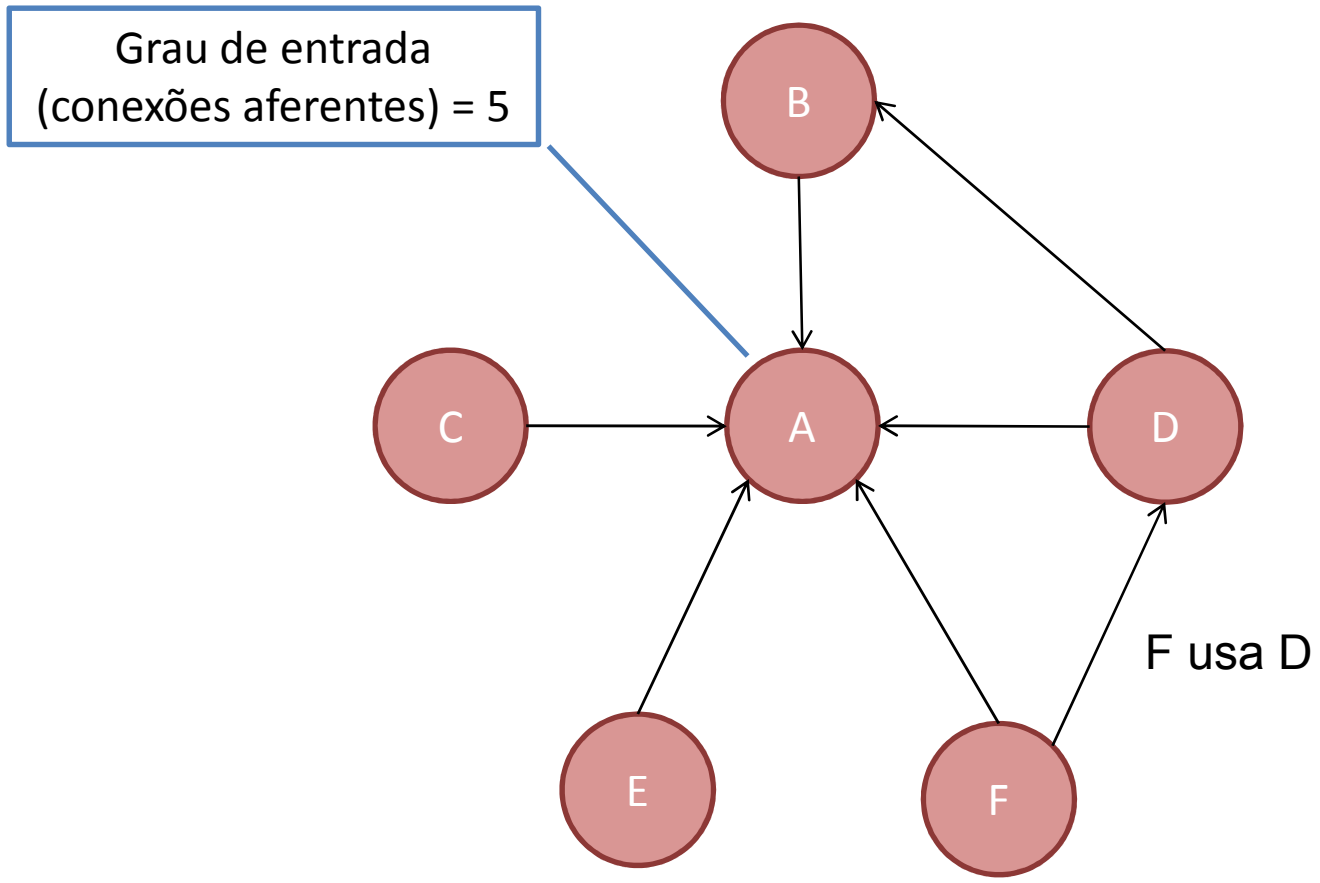


Um Estudo sobre Evolução da Estrutura de Software Orientado por Objetos

Evolução de Estrutura de Software OO

- Muitas pesquisas têm sido realizadas para caracterizar a evolução de software, especialmente em termos de tamanho e complexidade
  - Softwares crescem continuamente
  - Softwares têm complexidade crescente
- Recentemente, os conceitos de **Redes Complexas** têm sido aplicados para entender o comportamento e a natureza das estruturas de software

Evolução de Estrutura de Software OO



Rede de Software

Evolução de Estrutura de Software OO

- Os achados comuns desses trabalhos são:
  - Graus de entrada de vértices nas redes de módulos em um software seguem uma lei de potência
    - muitos vértices com grau pequeno, poucos vértices com grau muito alto
  - Tais redes parecem se adequar ao fenômeno *small-world*
    - Característica de redes nas quais os pares de vértices são conectados por um caminho curto
    - Esta propriedade é relacionada à facilidade de propagação de informação na rede

Evolução de Estrutura de Software OO

- Objetivos do estudo:
  - Investigar como a evolução das redes de software ocorre
    - Conectividade de software
    - Classes centrais (com muitas conexões aferentes)
    - Distância entre as classes de um software
    - Componente fortemente conectado
  - Identificar a figura da estrutura macroscópica das redes de software

Evolução de Estrutura de Software OO

- No estudo, a evolução de software é caracterizada por meio de um conjunto de métricas de software e de redes:
  - **Diâmetro**: maior caminho dentre os caminhos mínimos
  - **Grau de entrada**: número de conexões aferentes
  - **Coesão interna de classe**: métrica COR
  - **Densidade**: a / (n(n-1))
    - ***a*** é o número de arestas, ***n*** é o número de vértices
  - **Tamanho do maior componente fortemente conectado**
- Foram analisados 16 softwares abertos e um software comercial Java, em um total de 129 versões

Evolução de Estrutura de Software OO

- Principais conclusões do estudo :
  - A densidade do software diminui à medida que ele cresce.
    - A densidade é calculada como a métrica COF
    - Isso torna evidente que a métrica de densidade não deve ser tomada isoladamente para avaliar a complexidade e dificuldade de manutenção de software
  - O diâmetro da rede formada pelas classes do software é pequeno e cresce lentamente à medida que o software cresce

Evolução de Estrutura de Software OO

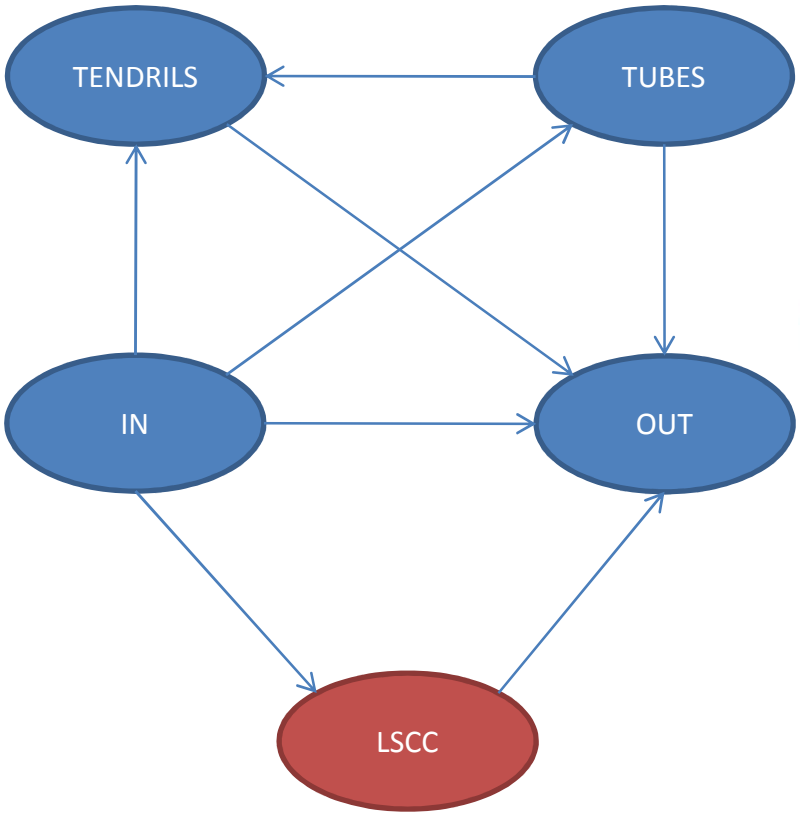
- Classes com maior número de conexões aferentes tendem a manter essa situação ao longo da vida do software
- O número de conexões aferentes dessas classes tende a crescer
- Tais classes são instáveis:
  - tamanho cresce (número de métodos públicos e de atributos públicos)
  - coesão interna diminui

➢ Conclusão: a prática usual parece ser manter as classes com alto grau de entrada, que já são grandes provedoras de serviços, nesta situação

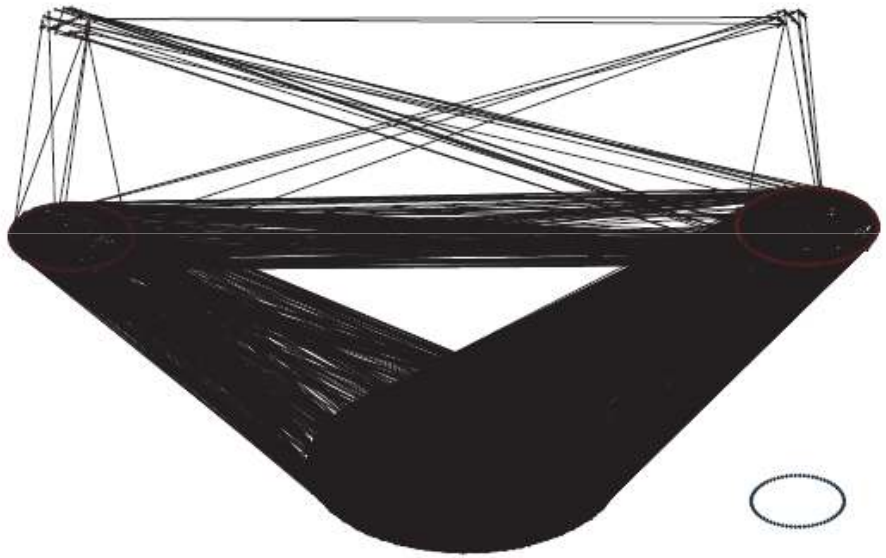
- incluindo ainda mais serviços nelas para que elas possam atender às novas classes

Evolução de Estrutura de Software OO

Little house - macroestrutura de software OO identificada



LSCC (*largest strongly connected component*)



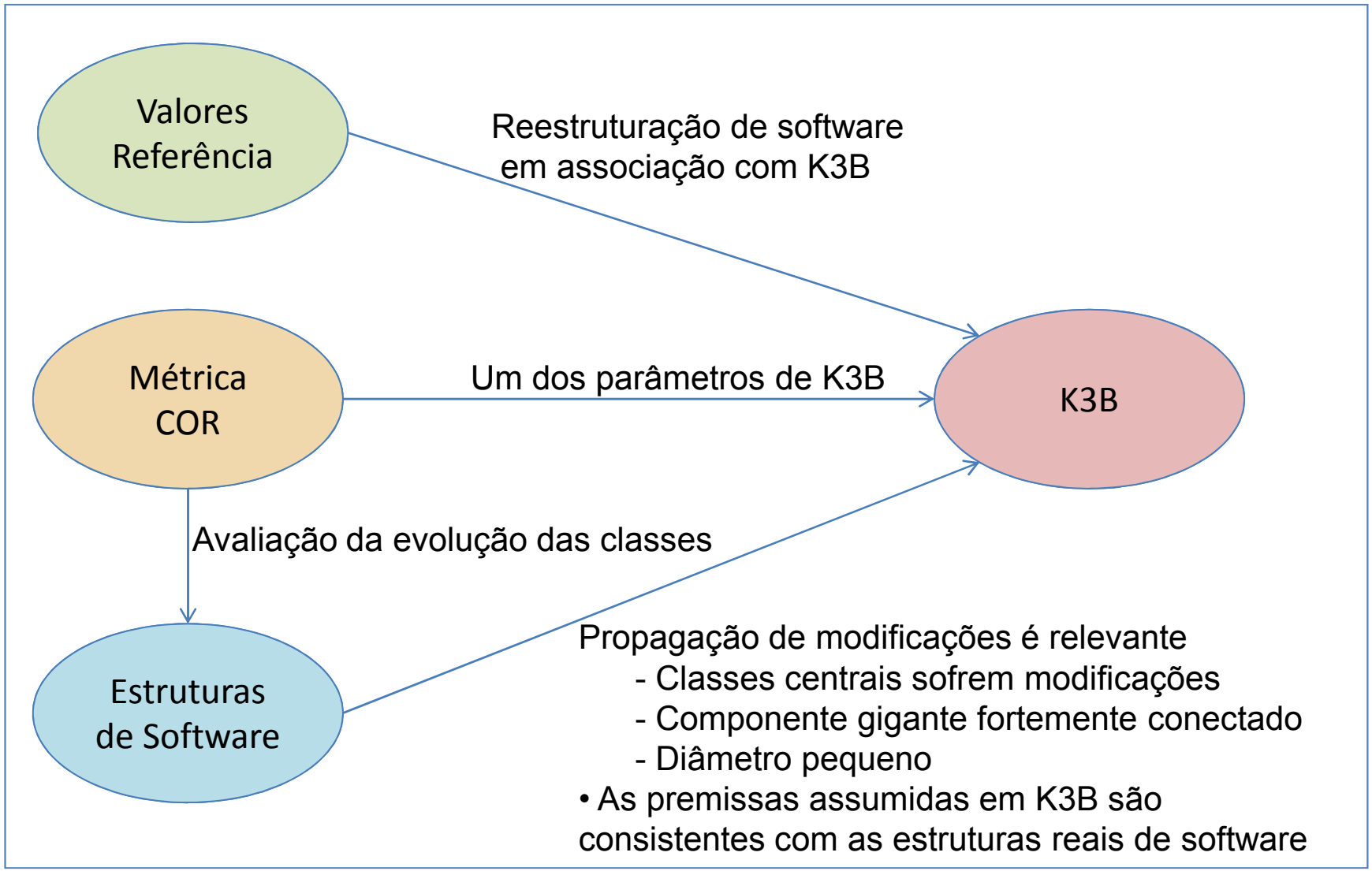
KolMafia 13.7

Evolução de Estrutura de Software OO

- O tamanho do LSCC cresce em número de classes
- A quantidade de classes em LSCC corresponde a um percentual significativo do total de classes do software
- Os resultados desse estudo evidenciam empiricamente que a propagação de modificações é um problema relevante
  - Redes de software têm diâmetro pequeno
  - Classes centrais sofrem modificações
  - Há um grande componente fortemente conectado nas redes de software

K3B – Um Modelo de Predição de Amplitude da Propagação de Modificações Contratuais em Software Orientado por Objetos

K3B


Valores
Referência
Reestruturação de software
em associação com K3B
Métrica
COR
Um dos parâmetros de K3B
K3B
Avaliação da evolução das classes
Estruturas
de Software
Propagação de modificações é relevante
- Classes centrais sofrem modificações
- Componente gigante fortemente conectado
- Diâmetro pequeno
• As premissas assumidas em K3B são
consistentes com as estruturas reais de software

K3B

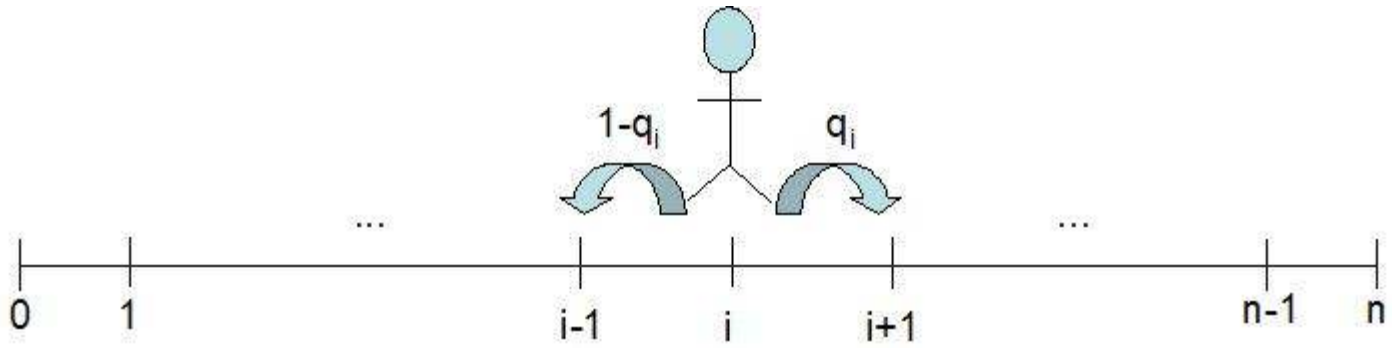
- Definição de K3B
  - Seja um software constituído por *n* módulos
  - *S* = (0, 1, 2, ..., n) denota o conjunto de estados possíveis para o processo de modificação
  - Cada estado corresponde ao número de módulos sofrendo modificações em um dado instante
  - Quando o número de módulos em modificação for igual a zero, diz-se que o processo alcançou estabilidade (*estado absorvente*)
  - Deseja-se saber ***o número médio de passos de modificações*** necessários para que um sistema chegue à estabilidade dado o seu estado, ou seja, o número de módulos em modificação

Definição de K3B

- Esse problema pode ser modelado por uma Cadeia de Markov
  - Se *i* é o estado presente, a probabilidade de se passar para os estados futuros depende somente do estado presente *i* e não dos estados anteriores a *i*
- A quantidade de módulos em modificação no instante futuro depende somente da quantidade de módulos em modificação no instante presente
- Dado que há ***i*** ($0 \leq i \leq n$) módulos em modificação em um instante
  - no próximo instante somente é possível ir para o estado i+1 ou para i-1

Definição de K3B

- No estado i, a probabilidade de se passar para o estado i+1 é q_i e de se passar para o estado i-1 é 1- q_i



Definição de K3B

- Utilizamos os termos
  - **conectividade**: é o percentual de conexões existentes de fato entre os módulos do software
    - **ϕ** é o parâmetro que indica a conectividade
  - **"taxa de contaminação":** quantidade de módulos nos quais será necessário realizar modificações em decorrência do número de módulos em modificação no estado corrente do sistema
    - **α** é o parâmetro que indica a taxa de contaminação
    - Deve ser um fator que quanto maior, maior também a taxa de contaminação
    - Candidato: grau de acoplamento entre os módulos

Definição de K3B

– **“taxa de melhora”:** quantidade de módulos que terão a modificação finalizada sem implicar na módulos de outros módulos

- **β** é o parâmetro que indica a taxa de melhora
- deve ser um fator que quanto maior, maior a probabilidade de modificações serem realizadas sem afetar outros módulos
- Candidato: grau de coesão interna dos módulos

Definição de K3B

- A partir do estado ***i***:
  - A probabilidade de se passar para o estado i+1 é proporcional
    - Ao número de módulos que não estão em modificação(n-i): quanto maior o número de módulos que não estão em modificação, maior a chance de que pelo menos um deles entre em modificação
    - Ao número de módulo que estão em manutenção (i): quanto maior o número de módulos em modificação, maior a chance de os outros entrarem em modificação em decorrência das modificações atuais
    - À taxa de contaminação (α)
    - À conectividade do software (ϕ)

$$\alpha \phi (n-i)i$$

Definição de K3B

- A probabilidade de se passar para o estado i-1 é proporcional
  - Ao número de módulo que estão em modificação(i): quanto maior o número de módulos em modificação, maior a chance de que pelo menos um finalize modificação no instante seguinte
  - À taxa de melhora (β) βi

- Como a soma de $p_{i,i+1}$ e $p_{i,i-1}$ deve ser igual a 1, obtêm-se as seguintes equações:

$$p_{i,i+1} = \frac{\alpha \phi (n-i)i}{\alpha \phi (n-i)i + \beta i} \quad \text{e} \quad p_{i,i-1} = \frac{\beta i}{\alpha \phi (n-i)i + \beta i}$$

Definição de K3B

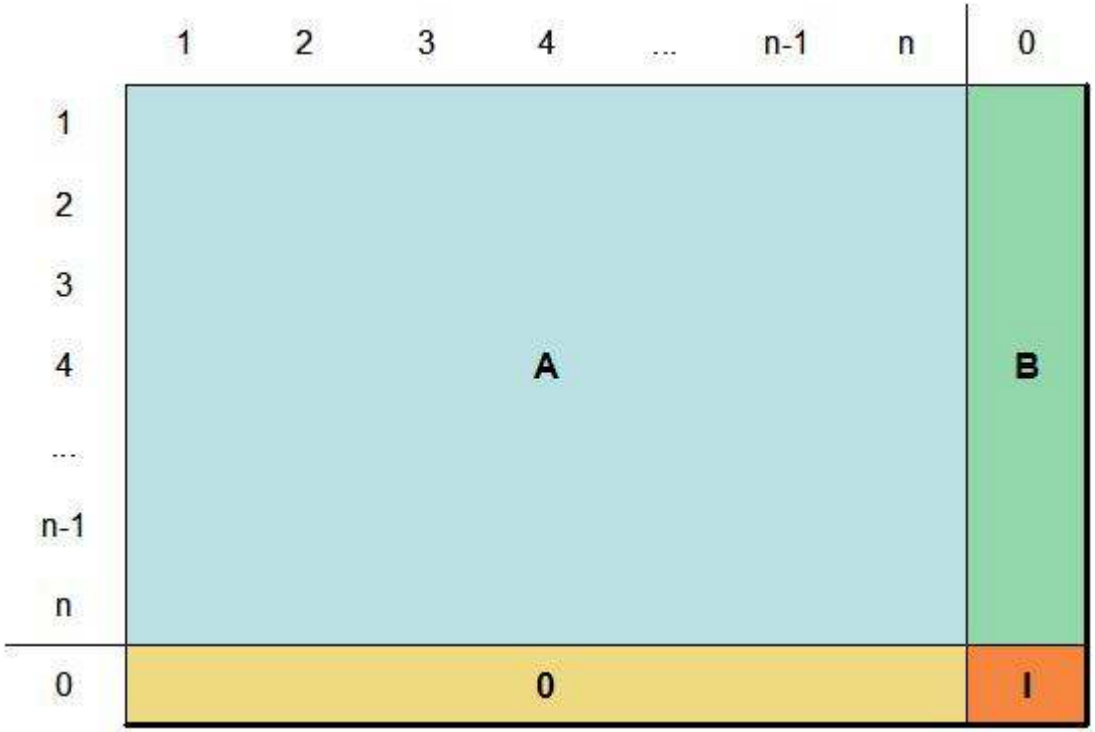
- O problema investigado consiste em
  - Determinar o número de passos necessários, $E(t_i)$, para que o sistema saia de um estado i e chegue ao *estado absorvente*
- Há um teorema que soluciona esta questão

  Demonstrado por Grinstead & Snell (1991), por exemplo
- Para que ele possa ser aplicado, é necessário que a matriz de probabilidades seja construída de forma que o estado absorvente seja a última linha e a última coluna da matriz

Definição de K3B

	1	2	3	4	...	n-1	n	0
1	**0**	q_1	0	0	0	0	0	$1- q_1$
2	$1-q_2$	**0**	q_2	0	0	0	0	0
3	0	$1-q_3$	**0**	q_3	0	0	0	0
4	0	0	$1-q_4$	**0**	...	0	0	0
...	...	...	...	...	...	...	...	...
n-1	0	0	0	0	$1-q_{n-1}$	**0**	q_{n-1}	0
n	0	0	0	0	0	1	**0**	0
0	0	0	0	0	0	0	0	1

Definição de K3B



- O teorema define que:

Seja $E(t_i)$ o número esperado de passos antes da cadeia ser absorvida, dado que a cadeia inicia no estado *i*, e seja E o vetor coluna cuja i-ésima entrada é $E(t_i)$. Então E = Nc, onde $N= (I-A)^{-1}$ e *c* é um vetor coluna no qual todas as entradas são iguais a 1.

Definição de K3B

- Desta forma:

$E(t_1)$
$E(t_2)$
$E(t_3)$
...
$E(t_n)$

$= (I - A)^{-1}$

1
1
1
1
1

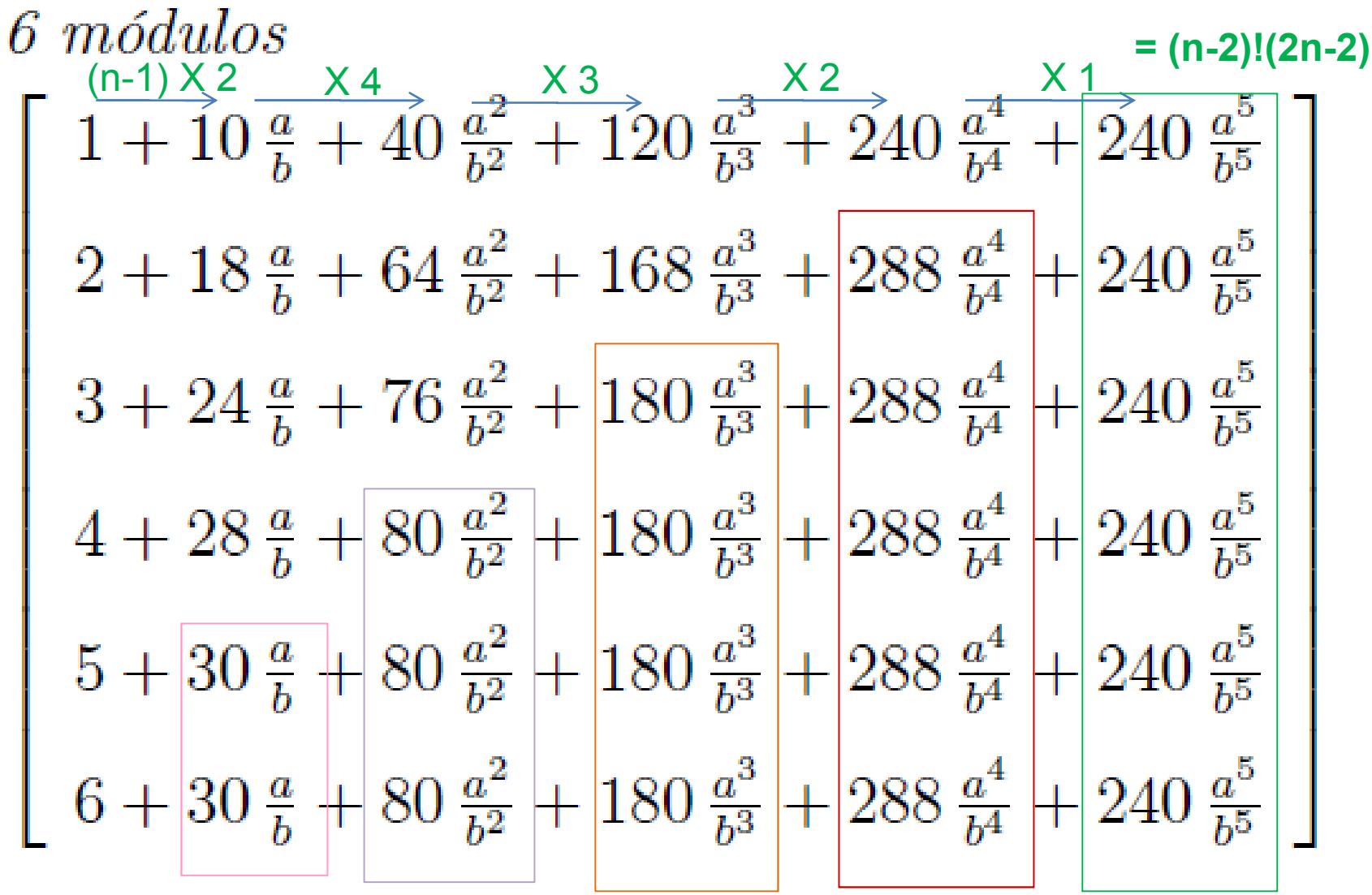
- Com auxílio do Matlab, o vetor E foi calculado para n Є (4,5,6,...,11,20,25)

K3B

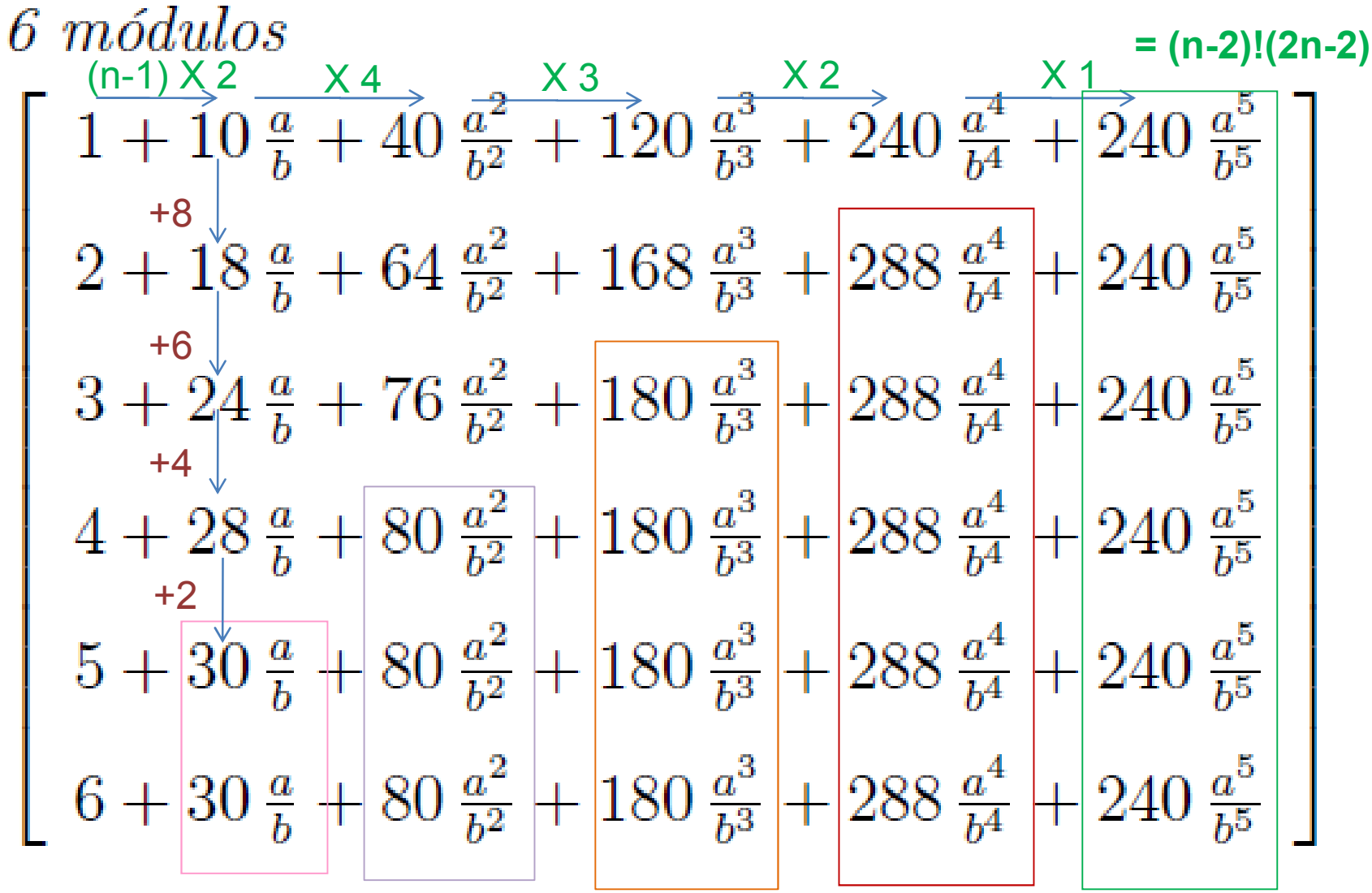
8 módulos

$$\begin{bmatrix}
1 + 14\,\frac{a}{b} + 84\,\frac{a^2}{b^2} + 420\,\frac{a^3}{b^3} + 1680\,\frac{a^4}{b^4} + 5040\,\frac{a^5}{b^5} + 10080\,\frac{a^6}{b^6} + 10080\,\frac{a^7}{b^7} \\
2 + 26\,\frac{a}{b} + 144\,\frac{a^2}{b^2} + 660\,\frac{a^3}{b^3} + 2400\,\frac{a^4}{b^4} + 6480\,\frac{a^5}{b^5} + 11520\,\frac{a^6}{b^6} + 10080\,\frac{a^7}{b^7} \\
3 + 36\,\frac{a}{b} + 184\,\frac{a^2}{b^2} + 780\,\frac{a^3}{b^3} + 2640\,\frac{a^4}{b^4} + 6720\,\frac{a^5}{b^5} + 11520\,\frac{a^6}{b^6} + 10080\,\frac{a^7}{b^7} \\
4 + 44\,\frac{a}{b} + 208\,\frac{a^2}{b^2} + 828\,\frac{a^3}{b^3} + 2688\,\frac{a^4}{b^4} + 6720\,\frac{a^5}{b^5} + 11520\,\frac{a^6}{b^6} + 10080\,\frac{a^7}{b^7} \\
5 + 50\,\frac{a}{b} + 220\,\frac{a^2}{b^2} + 840\,\frac{a^3}{b^3} + 2688\,\frac{a^4}{b^4} + 6720\,\frac{a^5}{b^5} + 11520\,\frac{a^6}{b^6} + 10080\,\frac{a^7}{b^7} \\
6 + 54\,\frac{a}{b} + 224\,\frac{a^2}{b^2} + 840\,\frac{a^3}{b^3} + 2688\,\frac{a^4}{b^4} + 6720\,\frac{a^5}{b^5} + 11520\,\frac{a^6}{b^6} + 10080\,\frac{a^7}{b^7} \\
7 + 56\,\frac{a}{b} + 224\,\frac{a^2}{b^2} + 840\,\frac{a^3}{b^3} + 2688\,\frac{a^4}{b^4} + 6720\,\frac{a^5}{b^5} + 11520\,\frac{a^6}{b^6} + 10080\,\frac{a^7}{b^7} \\
8 + 56\,\frac{a}{b} + 224\,\frac{a^2}{b^2} + 840\,\frac{a^3}{b^3} + 2688\,\frac{a^4}{b^4} + 6720\,\frac{a^5}{b^5} + 11520\,\frac{a^6}{b^6} + 10080\,\frac{a^7}{b^7}
\end{bmatrix}$$

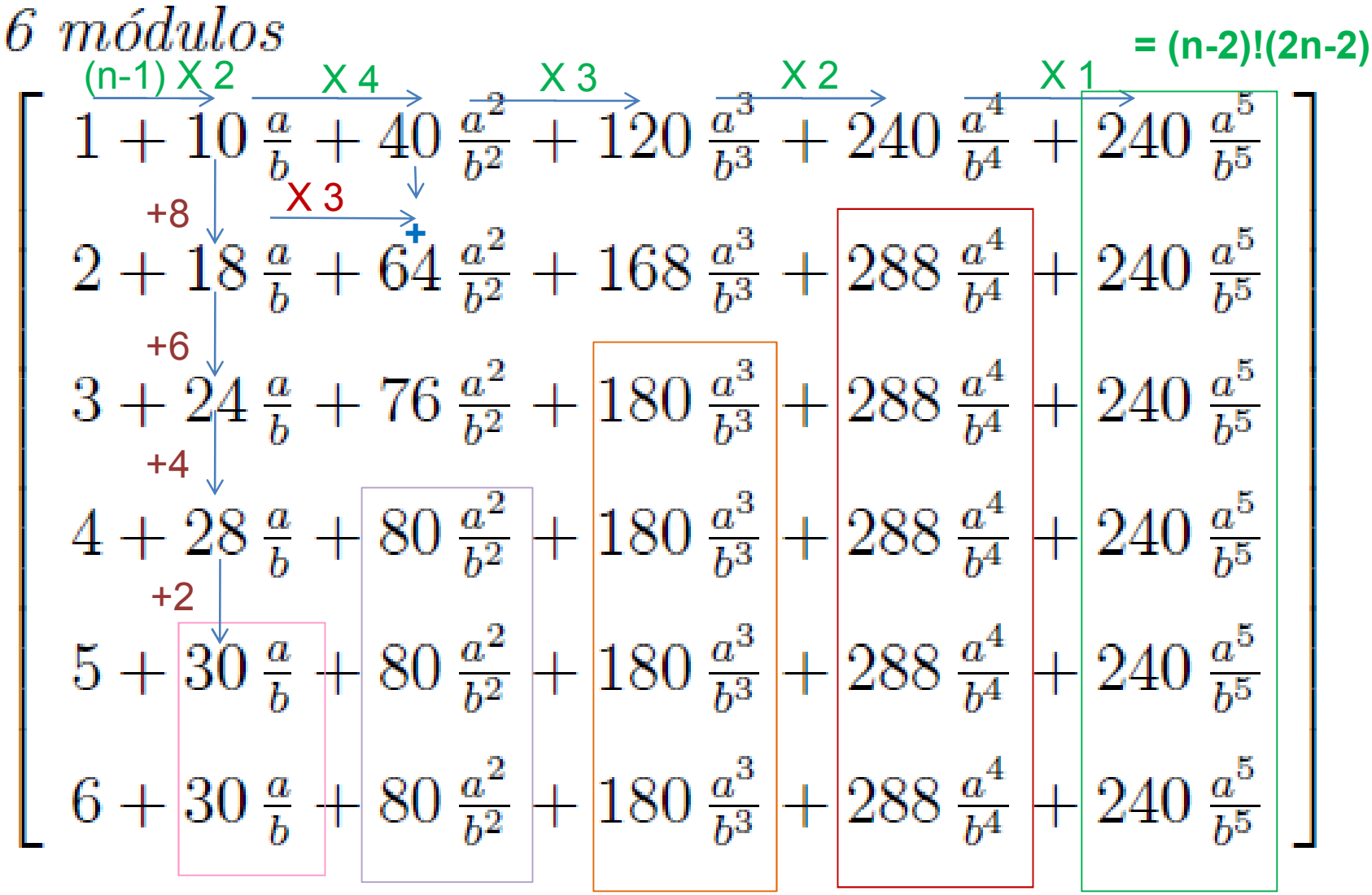
K3B


6 módulos
(n-1) X 2
X 4
X 3
X 2
X 1
= (n-2)!(2n-2)
$1 + 10\frac{a}{b} + 40\frac{a^2}{b^2} + 120\frac{a^3}{b^3} + 240\frac{a^4}{b^4} + 240\frac{a^5}{b^5}$
$2 + 18\frac{a}{b} + 64\frac{a^2}{b^2} + 168\frac{a^3}{b^3} + 288\frac{a^4}{b^4} + 240\frac{a^5}{b^5}$
$3 + 24\frac{a}{b} + 76\frac{a^2}{b^2} + 180\frac{a^3}{b^3} + 288\frac{a^4}{b^4} + 240\frac{a^5}{b^5}$
$4 + 28\frac{a}{b} + 80\frac{a^2}{b^2} + 180\frac{a^3}{b^3} + 288\frac{a^4}{b^4} + 240\frac{a^5}{b^5}$
$5 + 30\frac{a}{b} + 80\frac{a^2}{b^2} + 180\frac{a^3}{b^3} + 288\frac{a^4}{b^4} + 240\frac{a^5}{b^5}$
$6 + 30\frac{a}{b} + 80\frac{a^2}{b^2} + 180\frac{a^3}{b^3} + 288\frac{a^4}{b^4} + 240\frac{a^5}{b^5}$

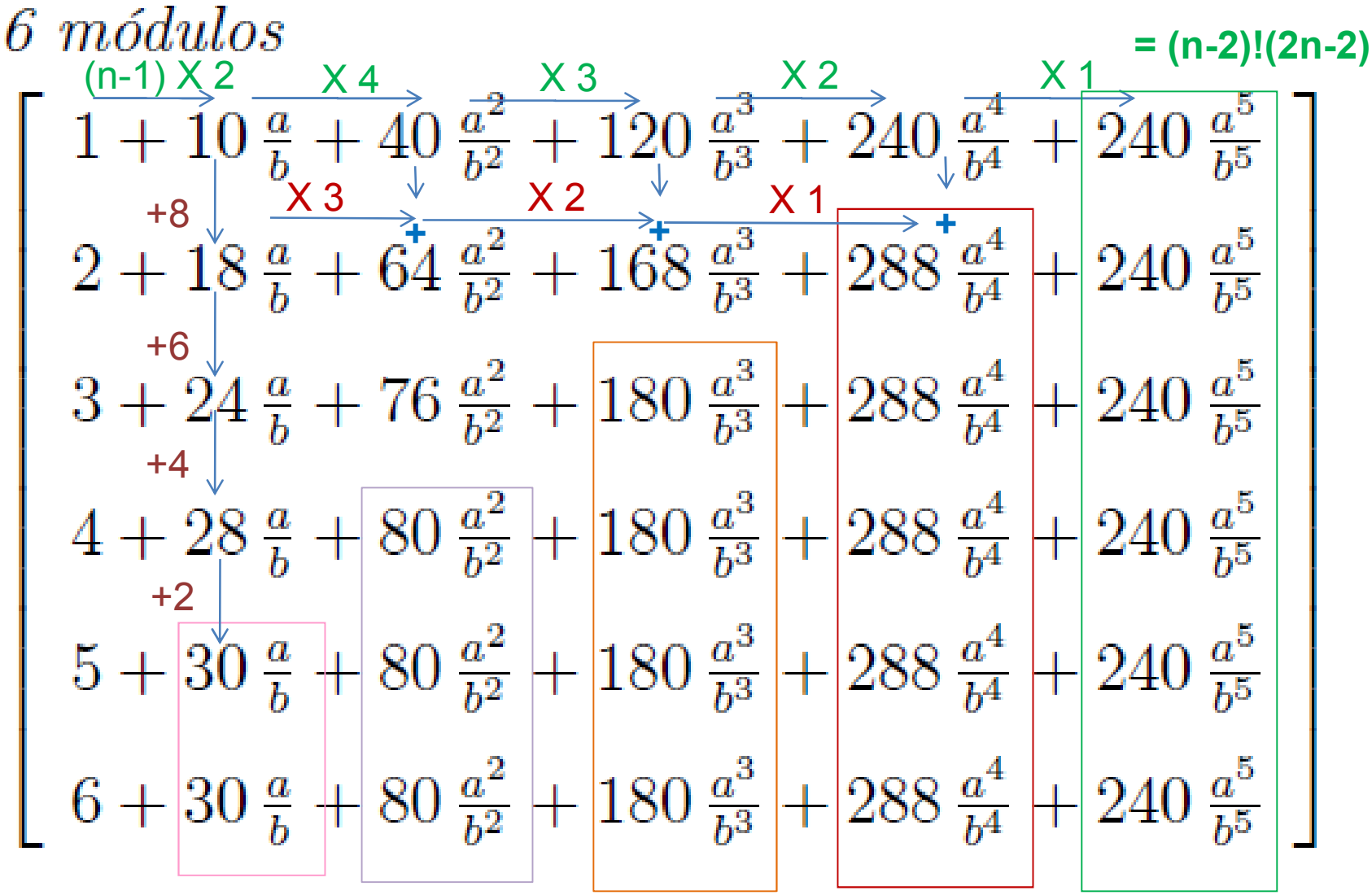
K3B


6 módulos
= (n-2)!(2n-2)
(n-1) X 2
X 4
X 3
X 2
X 1
$1 + 10\frac{a}{b} + 40\frac{a^2}{b^2} + 120\frac{a^3}{b^3} + 240\frac{a^4}{b^4} + 240\frac{a^5}{b^5}$
+8
$2 + 18\frac{a}{b} + 64\frac{a^2}{b^2} + 168\frac{a^3}{b^3} + 288\frac{a^4}{b^4} + 240\frac{a^5}{b^5}$
+6
$3 + 24\frac{a}{b} + 76\frac{a^2}{b^2} + 180\frac{a^3}{b^3} + 288\frac{a^4}{b^4} + 240\frac{a^5}{b^5}$
+4
$4 + 28\frac{a}{b} + 80\frac{a^2}{b^2} + 180\frac{a^3}{b^3} + 288\frac{a^4}{b^4} + 240\frac{a^5}{b^5}$
+2
$5 + 30\frac{a}{b} + 80\frac{a^2}{b^2} + 180\frac{a^3}{b^3} + 288\frac{a^4}{b^4} + 240\frac{a^5}{b^5}$
$6 + 30\frac{a}{b} + 80\frac{a^2}{b^2} + 180\frac{a^3}{b^3} + 288\frac{a^4}{b^4} + 240\frac{a^5}{b^5}$

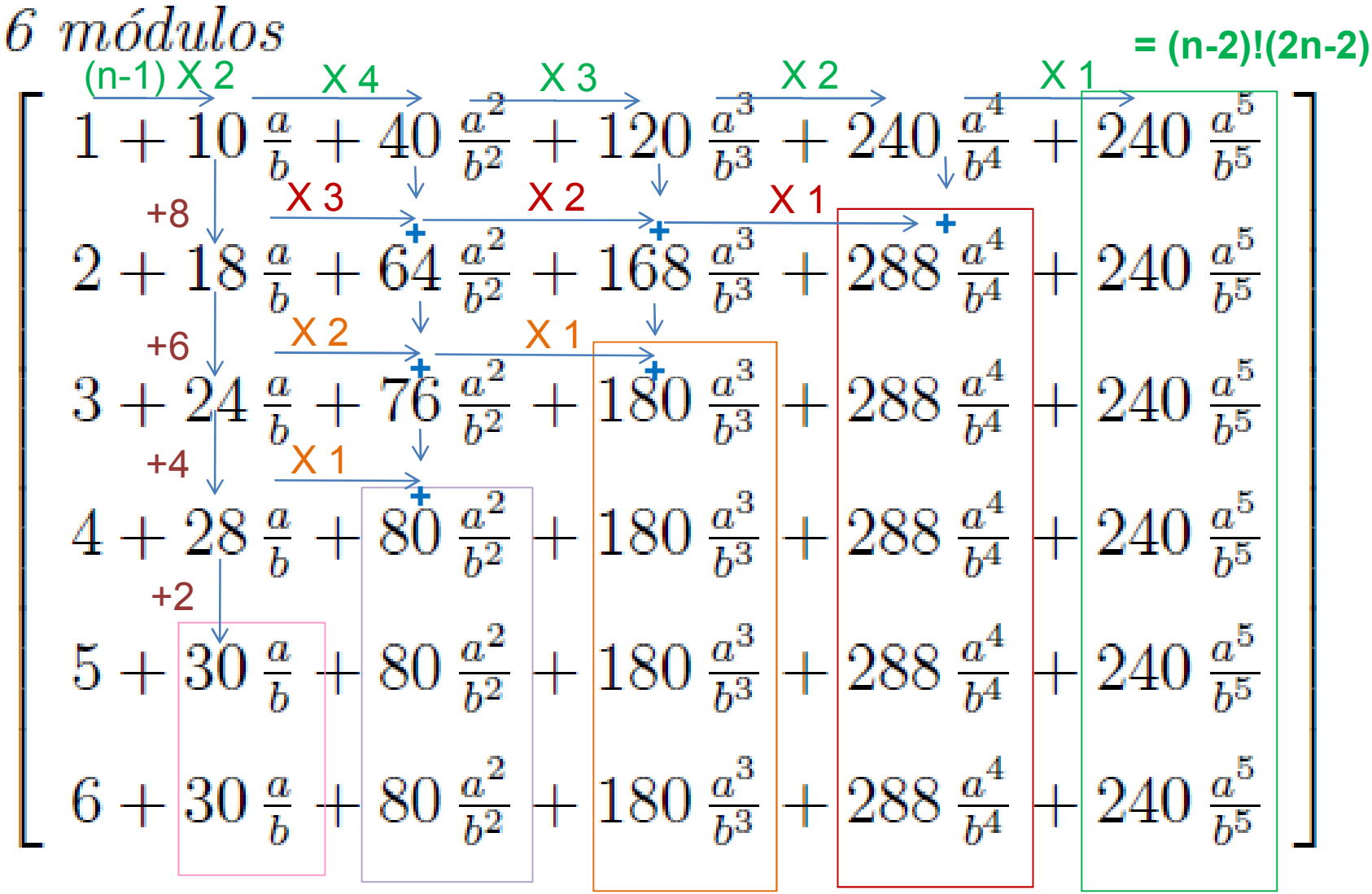
K3B


6 módulos
(n-1) X 2
X 4
X 3
X 2
X 1
= (n-2)!(2n-2)
X 3
+8
+6
+4
+2
$1 + 10\frac{a}{b} + 40\frac{a^2}{b^2} + 120\frac{a^3}{b^3} + 240\frac{a^4}{b^4} + 240\frac{a^5}{b^5}$
$2 + 18\frac{a}{b} + 64\frac{a^2}{b^2} + 168\frac{a^3}{b^3} + 288\frac{a^4}{b^4} + 240\frac{a^5}{b^5}$
$3 + 24\frac{a}{b} + 76\frac{a^2}{b^2} + 180\frac{a^3}{b^3} + 288\frac{a^4}{b^4} + 240\frac{a^5}{b^5}$
$4 + 28\frac{a}{b} + 80\frac{a^2}{b^2} + 180\frac{a^3}{b^3} + 288\frac{a^4}{b^4} + 240\frac{a^5}{b^5}$
$5 + 30\frac{a}{b} + 80\frac{a^2}{b^2} + 180\frac{a^3}{b^3} + 288\frac{a^4}{b^4} + 240\frac{a^5}{b^5}$
$6 + 30\frac{a}{b} + 80\frac{a^2}{b^2} + 180\frac{a^3}{b^3} + 288\frac{a^4}{b^4} + 240\frac{a^5}{b^5}$

K3B


6 módulos
= (n-2)!(2n-2)
(n-1) X 2
X 4
X 3
X 2
X 1
$1 + 10\frac{a}{b} + 40\frac{a^2}{b^2} + 120\frac{a^3}{b^3} + 240\frac{a^4}{b^4} + 240\frac{a^5}{b^5}$
+8
X 3
X 2
X 1
$2 + 18\frac{a}{b} + 64\frac{a^2}{b^2} + 168\frac{a^3}{b^3} + 288\frac{a^4}{b^4} + 240\frac{a^5}{b^5}$
+6
$3 + 24\frac{a}{b} + 76\frac{a^2}{b^2} + 180\frac{a^3}{b^3} + 288\frac{a^4}{b^4} + 240\frac{a^5}{b^5}$
+4
$4 + 28\frac{a}{b} + 80\frac{a^2}{b^2} + 180\frac{a^3}{b^3} + 288\frac{a^4}{b^4} + 240\frac{a^5}{b^5}$
+2
$5 + 30\frac{a}{b} + 80\frac{a^2}{b^2} + 180\frac{a^3}{b^3} + 288\frac{a^4}{b^4} + 240\frac{a^5}{b^5}$
$6 + 30\frac{a}{b} + 80\frac{a^2}{b^2} + 180\frac{a^3}{b^3} + 288\frac{a^4}{b^4} + 240\frac{a^5}{b^5}$

K3B


6 módulos
= (n-2)!(2n-2)
(n-1) X 2
X 4
X 3
X 2
X 1
$1 + 10\frac{a}{b} + 40\frac{a^2}{b^2} + 120\frac{a^3}{b^3} + 240\frac{a^4}{b^4} + 240\frac{a^5}{b^5}$
+8
X 3
X 2
X 1
$2 + 18\frac{a}{b} + 64\frac{a^2}{b^2} + 168\frac{a^3}{b^3} + 288\frac{a^4}{b^4} + 240\frac{a^5}{b^5}$
+6
X 2
X 1
$3 + 24\frac{a}{b} + 76\frac{a^2}{b^2} + 180\frac{a^3}{b^3} + 288\frac{a^4}{b^4} + 240\frac{a^5}{b^5}$
+4
X 1
$4 + 28\frac{a}{b} + 80\frac{a^2}{b^2} + 180\frac{a^3}{b^3} + 288\frac{a^4}{b^4} + 240\frac{a^5}{b^5}$
+2
$5 + 30\frac{a}{b} + 80\frac{a^2}{b^2} + 180\frac{a^3}{b^3} + 288\frac{a^4}{b^4} + 240\frac{a^5}{b^5}$
$6 + 30\frac{a}{b} + 80\frac{a^2}{b^2} + 180\frac{a^3}{b^3} + 288\frac{a^4}{b^4} + 240\frac{a^5}{b^5}$

K3B

O número de passos de modificações de um software com ***n*** módulos no qual ***i*** módulos são inicialmente modificados é dado pela seguinte equação

$$E(n,1) = 1 + 2(n-1)(n-2)!\,\frac{\alpha^{n-1}\phi^{n-1}}{\beta^{n-1}} + \sum_{k=1}^{k=n-2} \frac{2(n-1)(n-2)!}{k!}\,\frac{\alpha^{n-k-1}\phi^{n-k-1}}{\beta^{n-k-1}}$$

$$E(n,i) = 1 + E(n,i-1) + \sum_{k=i-1}^{k=n-2} 2(n-i)\,\frac{(n-i-1)!}{(k-i+1)!}\,\frac{\alpha^{n-k-1}\phi^{n-k-1}}{\beta^{n-k-1}}$$

K3B - Avaliação

- Decorre de K3B que:
  - O número de passos de modificações em software com alto grau de conectividade, com forte acoplamento entre os módulos e cujas classes têm baixa coesão é explosivo.
  - Mesmo em softwares de grande porte, se a conectividade, o acoplamento e a coesão têm bons níveis, a propagação de modificações é controlável.

K3B - Avaliação

- Foi desenvolvida uma ferramenta de simulação de modificações em programas Java.

- Dois tipos de modificações foram implementados:
  - Renomear método
  - **Modificar lista de parâmetro de método**

- O tipo de modificação analisado foi *modificar lista de parâmetros*
  - Para fins de simulação, este tipo de modificação é suficiente
  - É uma boa representante do efeito de propagação de modificações em software

K3B - Avaliação

- Para cada método do sistema, a ferramenta conta o número de passos de modificações resultantes
- Os valores de K3B foram comparados aos valores médios resultantes das simulações
- Foi realizada uma avaliação de K3B com 37 versões de 11 programas Java

K3B - Avaliação

Sim' = **m** K3B + **b**

Software	Versão	#classes	K3B(1)	K3B(2)	K3B(3)	Sim(1)	Sim(2)	Sim(3)	Correlação	m	b
Hibernate	3.0	956	2,33	4,66	6,99	2,67	5,34	8,01	0,99999997	1,15	0,00
	3.1	1118	2,38	4,76	7,14	2,73	5,47	8,01	0,99978741	1,11	0,12
JasperReports	0.4.0	242	1,75	3,49	5,23	2,73	5,47	8,20	0,99999998	1,57	-0,02
	1.0.0	574	1,88	3,77	5,65	2,79	5,58	8,38	1,00000000	1,48	0,00
	2.0.0	1104	1,97	3,94	5,90	2,82	5,65	8,79	0,99953185	1,52	-0,23
JavaGroups	1.0	415	3,01	6,01	8,99	3,73	7,44	11,14	0,99999996	1,24	0,00
	2.1.1	696	2,46	4,92	7,37	4,51	9,02	12,37	0,99633703	1,60	0,76
JML	10a1	171	2,61	5,20	7,77	2,33	4,66	6,98	0,99999954	0,90	-0,02
	10a2	186	2,18	4,35	6,51	2,01	4,01	6,02	0,99999818	0,93	-0,01
	10b1	203	2,42	4,82	7,21	2,13	4,27	6,41	0,99999960	0,89	-0,03
	10b3	218	2,50	4,99	7,47	2,16	4,33	6,49	0,99999949	0,87	-0,02
	10b4	270	2,74	5,48	8,20	2,59	5,18	7,77	0,99999889	0,95	-0,01

K3B - Avaliação

Software	Versão	K3B(1)	K3B(2)	K3B(3)	K3B(4)	Sim(1)	Sim(2)	Sim(3)	Sim(4)	Correlação	m	b
JML	10a1	2,61	5,20	7,77	10,33	2,33	4,66	6,98	9,29	0,999998535	0,9017	-0,0267
JSCH	0.1.14	2,17	4,32	6,44	8,54	2,3	4,60	6,89	9,17	0,999997453	1,0787	-0,0468
Squirrel	1.0	1,37	2,73	4,10	5,46	4,06	8,11	12,16	16,20	0,999999996	2,9689	-0,0041
KolMafia	1.0	2,47	4,93	7,37	9,79	5,53	11,02	16,46	21,85	0,999999882	2,2299	0,0243

Software	Versão	K3B(1)	K3B(2)	K3B(3)	K3B(4)	K3B(5)	Sim(1)	Sim(2)	Sim(3)	Sim(4)	Sim(5)	Correlação	m	b
Junit	3.4	1,41	2,81	4,21	5,60	6,99	1,67	3,32	4,96	6,58	8,19	0,999997486	1,1696	0,0270
MovieManager	1.6Beta	1,64	3,26	4,87	6,46	8,05	2,93	5,75	8,46	11,09	13,66	0,999905934	1,6736	0,2521
KoLMafia	0.2	1,65	3,27	4,86	6,46	8,02	2,83	5,69	8,58	11,49	14,41	0,999921943	1,8188	-0,2301

K3B - Avaliação

- Os resultados dessa avaliação:
  - Mostraram que há forte correlação entre os valores estimados por K3B e aqueles obtidos na simulação
    - Isso indica que K3B pode ser utilizado para estimar o valor simulado

K3B - Avaliação

- Mostraram que valor simulado pode ser obtido a partir do valor de K3B

Simulado = 1.7 * K3B

- Isso indica que K3B não só estima o valor simulado, como também há uma transformação genérica direta que mapeia K3B para o valor simulado, independente do software.

Conclusões

Conclusões

- O objetivo principal desta tese foi a definição de um modelo de predição de amplitude de propagação de modificações contratuais em software orientado por objetos
  - Número de passos de modificações quando ***i*** módulos são inicialmente modificados
- O modelo, denominado K3B
  - É uma expressão matemática na variável $(\alpha\phi)/\beta$
  - Φ: conectividade do software
  - α: taxa de contaminação do software
  - β : taxa de melhora do software

Conclusões

- O modelo K3B foi avaliado por meio de simulações de modificações em programas Java
- A simulação é um processo de custo alto
- O uso de K3B é uma alternativa eficiente para estimar propagação de modificações em software
- A avaliação de K3B mostrou que:
  - Há forte correlação entre K3B e as observações na simulação
  - É possível obter o valor simulado a partir de um ajuste de K3B por uma constante pequena e conhecida

Conclusões

- O modelo K3B tem as seguintes aplicações principais :
  - Auxílio a gerentes na alocação de recursos apropriados nas atividades de modificações
  - Comparação de softwares do ponto de vista de modificabilidade
  - Indicação da necessidade de reestruturação de software
    - Aumentar β e diminuir α
    - Utilizar métricas auxiliares, por exemplo *número de conexões aferentes, coesão, profundidade da árvore de herança, número de métodos públicos e número de atributos públicos*

Conclusões

- Os resultados desta tese geraram os seguintes artigos:
  - *Valores Referência de Métricas de Software Orientado por Objetos: SBES 2009, entre os cinco melhores artigos*
  - *Identifying Thresholds for Object-Oriented Software Metrics:* em fase final de revisão no *Journal of Systems and Software*
  - *Métrica de Coesão de Responsabilidade*: submetido a um congresso da área
  - *Software Evolution Characterization*: submetido a um periódico
  - *K3B - A Prediciting Model for Propagation of Contractual Modifications in Software*: a ser submetido a um periódico

Conclusões

- Trabalhos futuros
  - Explorar o modelo *little house* e suas implicações
  - Estender a implementação de K3B para outras linguagens de programação e para modelos UML
  - A criação de um ambiente integrado que permita, por exemplo:
    - Rastreabilidade de requisitos em classes
    - Seleção dos requisitos a serem modificados e correspondente cálculo de K3B
    - Avaliação de métricas de software auxiliares e seus respectivos valores referência

Fim

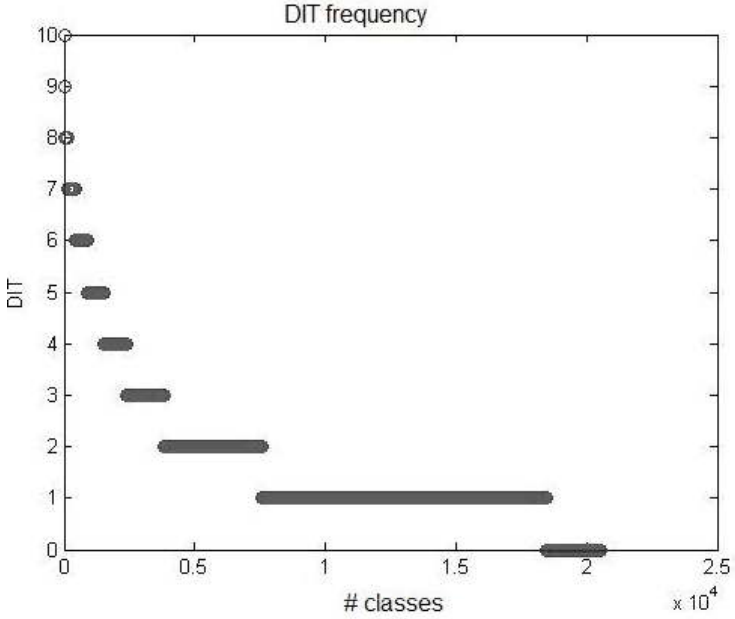
Valores Referência - Softwares

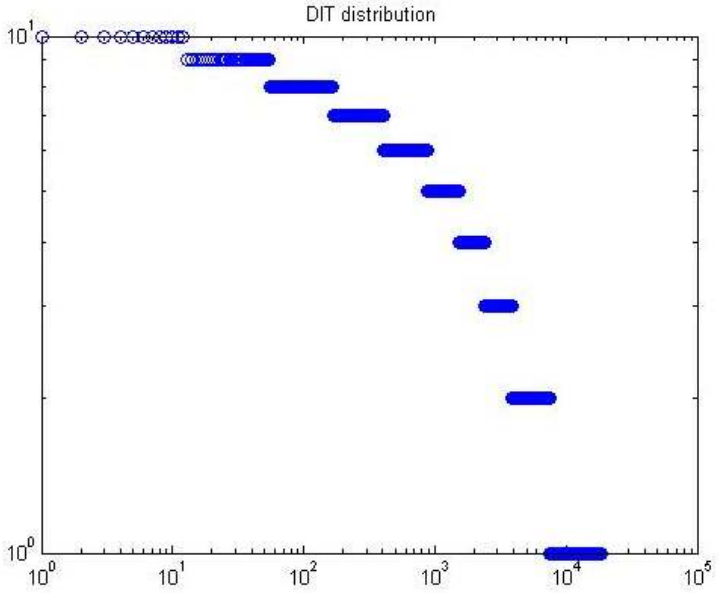
Domínio	Software	Tipo	#Classes	#Conexões	COF
Clustering	Essence	framework	182	543	0,016
	Gridsim	tool	214	774	0,017
	JavaGroups	tool	1061	3807	0,003
	Prevayler	library	90	137	0,017
	Super (Acelet-Scheduler)	tool	246	1085	0,018
Database	DBUnit	framework	289	911	0,011
	ERMaster	tool	569	2187	0,007
	Hibernate	framework	1359	5199	0,003
Desktop	Facilitator	tool	2234	6565	0,001
	Java Gui Builder	tool	60	126	0,036
	Java X11 Library	library	318	1146	0,011
	J-Pilot	tool	142	367	0,018
	Scope	framework	214	535	0,012
Development	Code Generation Library	library	226	662	0,013
	DrJava	tool	2766	9684	0,001
	Find Bugs	tool	1019	3108	0,003
	Jasper Reports	library	1233	5610	0,004
	Junit	framework	154	353	0,015
	Spring	framework	2116	7069	0,002
	BCEL	library	373	2111	0,015

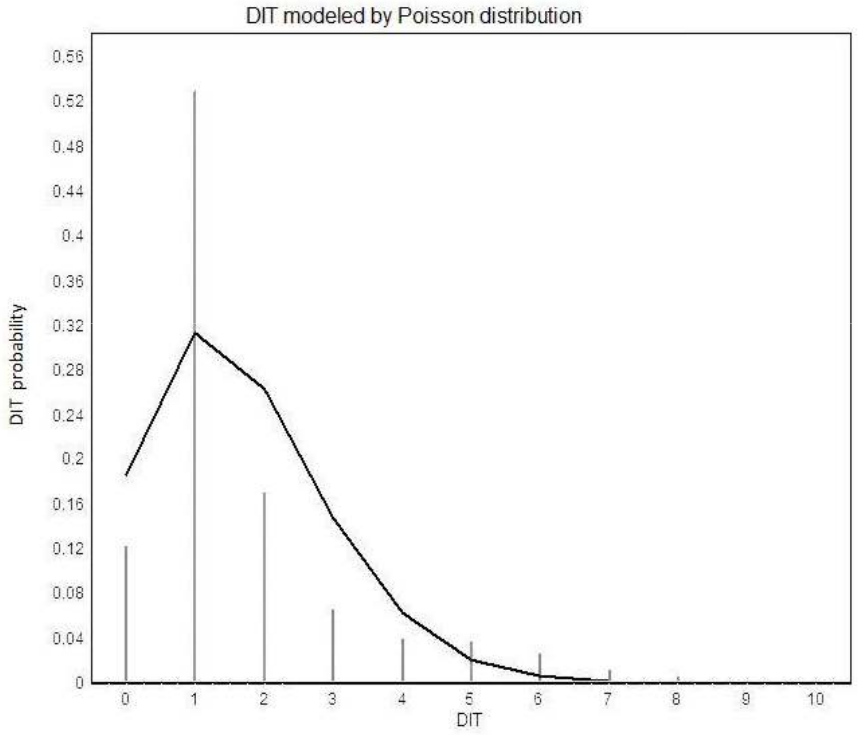
Valores Referência - Softwares

Domínio	Software	Tipo	#Classes	#Conexões	COF
Enterprise	Liferay	framework	14	14	0,077
	Talend	tool	2779	3567	0,000822
	uEngine BPM	framework	708	1774	0,004
	YAWL	tool	382	1186	0,008
Financial	JMoney	tool	193	424	0,019
Games	JSpaceConquest	tool	150	424	0,019
	KoLmafia	tool	810	5106	0,008
	Robocode	tool	213	738	0,016
Hardware	Jcapi	library	21	61	0,145
	LibUSBJava	library	35	90	0,076
	ServoMaster	library	55	117	0,039
Multimedia	CDK	library	3586	14711	0,001
	JPedal	tool	539	1533	0,005
	Pamguard	tool	1503	5267	0,002
Networking	BlueCove	library	142	461	0,023
	DHCP4Java	library	18	29	0,095
	jSLP	library	42	156	0,091
	WiKID Strong Authentication	library	50	27	0,011
Security	JSch	library	110	226	0,022
	OODVS	library	171	325	0,011

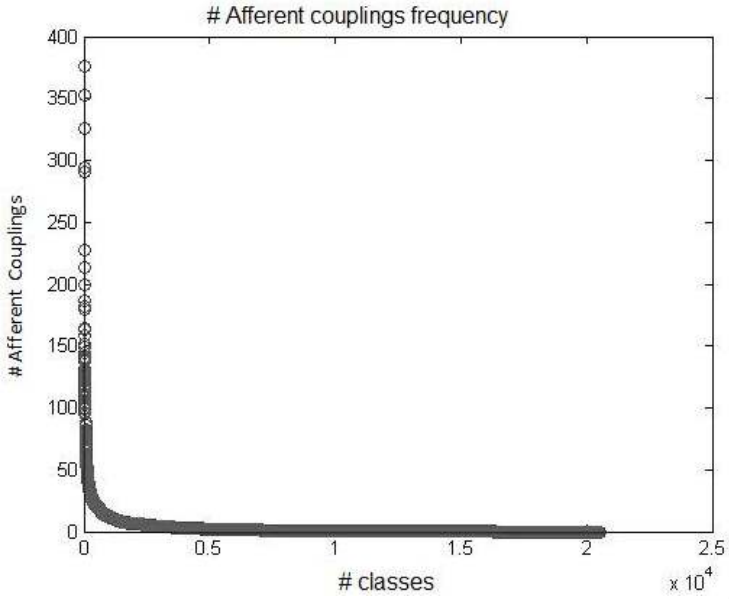
Valores Referência - Softwares

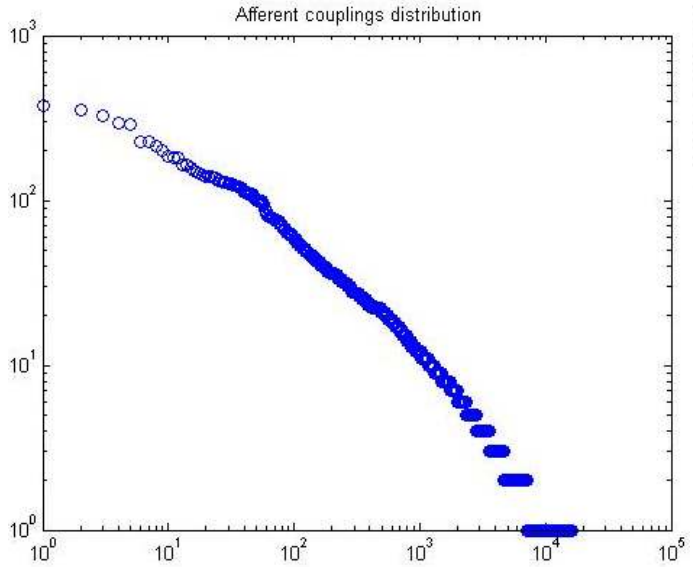


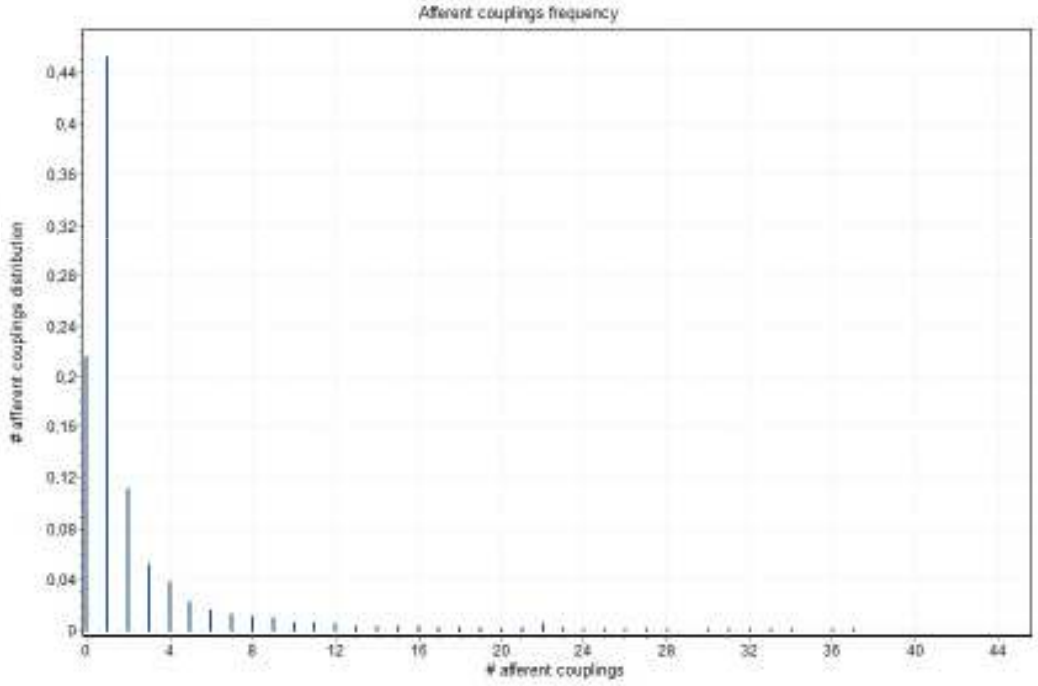




Valores Referência - Softwares







Evolução de Softwares

Software	Categoria	downloads/ semana	idade	#classes	#versões	#versões analisadas
JEdit	Text Editor	9.138	2001 a 2009	377 a 1124	13	13
Dr Java	Development	3.837	2002 a 2009	596 a 3692	10	10
Java Groups	Cooperation	465	2003 a 2009	696 a 1137	40	13
KoL Mafia	Game	1.007	2004 a 2009	39 a 1109	13	13
DBUnit	Database	448	2002 a 2009	198 a 369	25	5
FreeCol	Game	7.452	2003 a 2010	112 a 5902	27	5
JasperReports	Development	5.542	2001 a 2010	525 a 5304	50	5
JGNash	Financial	822	2002 a 2010	782 a 3603	40	5
Java msn library	Communication	271	2004 a 2010	494 a 872	10	5
Jsch	Security	2.304	2004 a 2009	202 a 271	29	5
JUnit	Development	1.834	2000 a 2009	78 a 230	18	5
Logisim	Education	1.590	2005 a 2009	908 a 1185	28	5
MeD's Movie Manager	Storage	1.169	2003 a 2010	64 a 517	60	5
Phex	Network	1.084	2001 a 2009	393 a 1352	26	5
Squirrel sql	Database	7.270	2006 a 2010	424 a 1223	26	5
Hibernate	Database	12.906	2004 a 2010	956 a 2446	53	5
Commercial - frontier layer	Commercial application	-	2005 - 2010	1100 a 1246	10	10
Commercial - model layer	Commercial application	-	2005 - 2010	3343 a 4031	10	10

Evolução de Softwares

Software	Versão	Classes	Conexões	Dens.	Diâm.
DBUnit	2.0	198	429	0,011	9
	2.2.1	289	666	0,008	11
	2.4.0	332	769	0,007	13
	2.4.4	347	780	0,006	17
	2.4.7	369	815	0,006	16
FreeCol	0.1.0	44	112	0,059	5
	0.5.0	416	1899	0,011	12
	0.6.0	611	2609	0,007	11
	0.8.0	927	5150	0,006	13
	0.9.2	1087	5902	0,005	14
Jasper	0.4.0	242	525	0,009	8
Reports	1.0.0	574	1316	0,004	9
	2.0.0	1104	2435	0,002	13
	3.0.0	1233	3038	0,002	13
	3.7.1	1629	5304	0,002	13
JGNash	1.10.0	743	2757	0,005	16
	1.11.1	782	2443	0,004	17
	1.50.0	942	2659	0,003	12
	2.00.0	2716	7374	0,001	24
	2.20.0	3603	12978	0,001	24
Java	10a1	171	494	0,017	10
msn	10a2	186	516	0,015	7
library	10b1	203	615	0,015	7
	10b2	218	662	0,014	9
	10b3	270	872	0,012	9

Software	Versão	Classes	Conexões	Dens.	Diâm.
LogSim	2.0.0	908	3294	0,004	13
	2.1.0	993	3940	0,004	14
	2.1.5	1018	4141	0,004	14
	2.2.0	1054	4439	0,004	14
	2.3.3	1185	4609	0,003	14
MeD's	1.6	64	149	0,037	6
Movie	1.7	73	168	0,032	6
Manager	2.0	517	1067	0,004	10
	2.8	458	1465	0,007	12
	2.9.13	608	1845	0,005	13
Phex	0.6	393	1078	0,007	8
	2.0.0	897	3215	0,004	18
	2.8.0	1205	4352	0,003	16
	3.0.0	1419	6036	0,003	19
	3.4.2	1352	5480	0,003	20
Squirrel	1.0	424	717	0,004	15
-sql	2.0	729	1592	0,003	13
	2.6	940	1765	0,002	14
	3.0	1134	2570	0,002	16
	3.1	1223	2989	0,002	16
JSch	0.1.1.4	80	202	0,032	4
	0.1.20	83	204	0,028	5
	0.1.26	94	210	0,024	5
	0.1.34	109	271	0,023	5
	10.1.42	117	385	0,02	5

Evolução de Softwares

Software	Versão	Classes	Conexões	Dens.	Diâm.
JUnit	3.4	78	138	0,023	5
	3.8	101	182	0,018	6
	4.0	92	197	0,02	6
	4.5	188	352	0,01	8
	4.8.1	230	421	0,008	10
Java	2.2	696	1935	0,004	10
Groups	2.2.1	849	2880	0,004	10
	2.2.5	829	2059	0,003	10
	2.2.6	832	2074	0,003	10
	2.2.7	857	2201	0,003	10
	2.2.8	810	2621	0,004	8
	2.2.9	922	2621	0,003	8
	2.3	959	2756	0,003	9
	2.4.1	1013	3075	0,003	7
	2.5.1	967	3736	0,004	8
	2.6.1	1012	3639	0,003	8
	2.7.0	1041	3688	0,003	11
	2.8.0	1137	3875	0,003	9
KolMafia	0.2	39	83	0,056	7
	0.4	75	222	0,04	9
	1.0	143	508	0,025	10
	2.0	191	726	0,02	11
	4.0	342	1399	0,012	12
	5.0	334	1780	0,016	11
	6.0	388	2102	0,014	10
	7.0	498	2970	0,012	12
	9.0	616	3410	0,009	11
	10.0	708	4004	0,008	13
	11.0	757	4578	0,008	14
	12.0	772	5357	0,009	12
	13.7	1109	7373	0,006	13

Software	Versão	Classes	Conexões	Dens.	Diâm.
Hibernate	3.0	956	2739	0,003	19
	3.1	1118	3746	0,003	20
	3.2	1302	4102	0,002	23
	3.3.0	1690	5707	0,002	21
	3.5.1	2446	5980	0,001	21
JEdit	2.4	377	1192	0,009	8
	2.5	422	1474	0,008	8
	3.1	426	1595	0,009	8
	3.2	449	1672	0,008	8
	4.0	554	2059	0,007	9
	4.1	618	2393	0,006	12
	4.1-8	646	2550	0,006	12
	4.2	805	3255	0,005	10
	4.3	810	3276	0,005	10
	4.3.4	867	3444	0,005	10
	4.3.9	954	3671	0,004	10
	4.3.13	1008	3885	0,004	13
	4.3.18	1124	4261	0,003	12
DrJava	1011	596	1773	0,005	10
	2148	1064	3393	0,003	14
	1826	1108	3680	0,003	12
	2304	1512	4569	0,002	18
	2332	1622	5259	0,002	19
	1750	2036	8287	0,002	21
	1406	2187	9562	0,002	23
	1942	3003	9732	0,001	17
	r4592	3421	117000	0,001	14
	r4756	3692	13627	0,001	16

Evolução de Softwares

FreeCol	0.5.0						0.9.2						Crescimento	
Classe	**Conex.**	**LCOM**	**COR**	**DIT**	**Atr.**	**Mét.**	**Conex.**	**LCOM**	**COR**	**DIT**	**Atr.**	**Mét.**	**#Mét.**	**Vezes**
net.sf.freecol.client.FreeColClient	84	481	0,17	1	4	31	214	955	0,25	1	1	49	18	1,00
net.sf.freecol.common.model.Unit	84	9145	0,13	0	75	150	148	11189	0,17	0	2	172	22	1,15
net.sf.freecol.client.gui.Canvas	80	1909	0,33	5	34	79	208	2359	0,25	6	0	92	13	1,16
net.sf.freecol.common.model.Player	78	5812	0,14	0	38	128	158	11940	0,17	0	5	167	39	1,30
net.sf.freecol.common.model.Game	67	742	0,33	0	3	45	131	1053	0,2	0	0	49	4	1,09
net.sf.freecol.common.model.Tile	61	2453	0,14	0	23	77	131	3085	0,33	0	1	97	20	1,26
net.sf.freecol.client.gui.i18n.Messages	59	9	0,5	1	3	4	174	58	0,33	1	3	26	22	6,50
net.sf.freecol.client.gui.panel.FreeColDialog	45	149	0,17	0	11	20	49	74	0,2	0	0	16	-4	0,80
net.sf.freecol.common.model.Map	42	407	0,13	0	21	41	63	537	0,33	0	2	52	11	1,27
net.sf.freecol.client.control.InGameController	40	0	0,5	1	3	45	69	0	1	1	0	65	20	1,44
												Média=	16,11	1,73

Evolução de Softwares

Hibernate	3.0						3.5.1						Crescimento	
Classe	**Conex.**	**LCOM**	**COR**	**DIT**	**Atr.**	**Mét.**	**Conex.**	**LCOM**	**COR**	**DIT**	**Atr.**	**Mét.**	**#Mét.**	**Vezes**
org.hibernate.HibernateException	86	0	1	0	0	3	174	0	1	4	0	3	0	1,00
org.hibernate.util.StringHelper	81	378	0,04	1	0	32	139	666	0,04	1	1	44	12	1,38
org.hibernate.dialect.Dialect	58	2467	0,02	1	2	69	97	7596	0,01	1	4	123	54	1,78
org.hibernate.MappingException	49	0	1	0	0	3	87	0	1	0	0	3	0	1,00
org.hibernate.util.ArrayHelper	47	227	0,05	1	6	25	59	274	0,05	1	8	33	8	1,32
org.hibernate.Hibernate	45	91	0,08	1	27	17	68	105	0,08	1	34	25	8	1,47
org.hibernate.util.ReflectHelper	42	139	0,14	1	2	14	73	151	0,2	1	3	17	3	1,21
org.hibernate.AssertionFailure	38	1	0,5	0	0	2	65	1	0,5	4	0	2	0	1,00
												Média=	10,71	1,21

Evolução de Softwares

JasperReports	0.4.0						3.7.1						Crescimento	
Classe	**Conex.**	**LCOM**	**COR**	**DIT**	**Atr.**	**Mét.**	**Conex.**	**LCOM**	**COR**	**DIT**	**Atr.**	**Mét.**	**#Mét.**	**Vezes**
net.sf.jasperreports.engine.xml.JRBaseFactory	94	4	0,33	1	0	5	153	0	1	1	0	3	-2	0,60
net.sf.jasperreports.engine.JRExpressionCollector	55	13	1	1	0	26	86	195	1	1	0	54	28	2,08
net.sf.jasperreports.engine.JRException	49	0	1	3	0	7	90	0	1	3	0	3	-4	0,43
net.sf.jasperreports.engine.fill.JRFillVariable	47	177	1	1	3	26	14	278	0,5	1	0	32	6	1,23
net.sf.jasperreports.engine.base.JRBaseObjectFactory	42	1033	1	0	0	35	72	3395	1	0	0	67	32	1,91
net.sf.jasperreports.engine.JRAbstractObjectFactory	28	0	1	1	0	29	5	3	0,5	1	0	27	-2	0,93
net.sf.jasperreports.engine.xml.JRXmlConstants	27	442	0,33	1	5	30	51	1046	0,33	1	390	46	16	1,53
net.sf.jasperreports.engine.xml.JRXmlWriter	26	0	1	1	0	30	7	0	0,5	0	1	57	27	1,90
net.sf.jasperreports.engine.fill.JRFillObjectFactory	25	657	1	0	0	33	39	2344	0,33	0	0	64	31	1,94
net.sf.jasperreports.engine.fill.AbstractValueProvider	25	2	0,5	1	0	4	33	2	0,5	1	0	4	0	1,00
												Média=	10,57	1,24

Evolução de Softwares

DrJava	1013						r4756						Crescimento	
Classe	**Conex.**	**LCOM**	**COR**	**DIT**	**Atr.**	**Mét.**	**Conex.**	**LCOM**	**COR**	**DIT**	**Atr.**	**Mét.**	**#Mét.**	**Vezes**
koala.dynamicjava.tree.Node	158	91	1	1	5	19	158	32	0,5	1	1	17	-2	0,89
koala.dynamicjava.tree.Expression	78	0	1	2	0	0	130	0	1	2	0	4	4	4,00
koala.dynamicjava.tree.BinaryExpression	36	0	1	3	2	4	34	3	1	3	2	5	1	1,25
edu.rice.cs.drjava.model.GlobalModelListener	32	0	1	1	1	13	3	0	1	1	0	29	16	2,23
koala.dynamicjava.tree.PrimaryExpression	29	0	1	3	0	0	37	0	1	3	0	0	0	1,00
koala.dynamicjava.tree.ExpressionStatement	26	0	1	1	0	0	4	26	0,33	3	0	8	8	8,00
edu.rice.cs.drjava.ui.MainFrame	25	234	0,14	6	0	7	324	24322	0,05	7	9	90	83	12,86
koala.dynamicjava.tree.ExpressionContainer	20	0	1	1	1	2	24	0	1	1	1	2	0	1,00
koala.dynamicjava.tree.Statement	16	0	1	2	0	0	27	0	1	2	0	4	4	4,00
koala.dynamicjava.interpreter.error.ExecutionError	16	4	0,5	3	2	9	5	4	0,5	3	2	9	0	1,00
												Média=	11,4	3,62

Evolução de Softwares

KolMafia	2.0						13.7						Crescimento	
Classe	**Conex.**	**LCOM**	**COR**	**DIT**	**Atr.**	**Mét.**	**Conex.**	**LCOM**	**COR**	**DIT**	**Atr.**	**Mét.**	**#Mét.**	**Vezes**
net.sourceforge.kolmafia.KoLmafia	69	272	0,33	1	0	30	264	3309	0,05	1	17	78	48	2,60
net.java.dev.spellcast.utilities.LockableListModel	34	241	0,5	2	0	37	151	488	0,33	2	0	49	12	1,32
net.sourceforge.kolmafia.AdventureResult	26	14	1	1	8	13	188	125	0,33	1	16	38	25	2,92
net.java.dev.spellcast.utilities.ActionVerifyPanel	26	18	0,33	5	0	3	39	5	0,5	6	0	17	14	5,67
net.sourceforge.kolmafia.KoLCharacter	20	2354	0,08	1	0	69	187	20862	0,13	2	13	228	159	3,30
												Média=	51,60	3,16

Evolução de Softwares – Crescimento de LSCC

Software	LSCC – versão inicial (# classes)	LSCC – versão inicial (%)	LSCC – versão final (# classes)	LSCC – versão final (%)
Jsch	16	20	16	14
LogSim	289	32	303	25
Jml	17	10	27	10
JavaGroups	9	2	13	1
DBUnit	16	8	24	7
Hibernate	100	10	477	20
Squirel	40	10	579	47
Junit	22	28	9	4
Jedit	69	20	395	35
Phex	165	41	403	29
Jgnash	276	35	322	11
DrJava	43	2	968	26
Jasper	20	8	150	9
MovieManager	50	78	92	18
FreeCol	5	11	758	70
KolMafia	15	38	748	67