HERON DE SOUZA MARQUES

PEIXE-DIPNOICO: UM FRAMEWORK PARA PREDIÇÃO DE MEDIDAS ESTRUTURAIS DE SOFTWARE EM SÉRIES TEMPORAIS

Dissertação apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense, como requisito parcial para obtenção do Grau de Mestre. Área de Concentração: Engenharia de Software.

Orientador: Prof. Dr. LEONARDO GRESTA PAULINO MURTA
Coorientador: Prof. Dr. ALEXANDRE PLASTINO DE CARVALHO

Niterói
2013

Ficha Catalográfica – Esta página deve ser removida na versão a ser entregue para a banca, mas deve ser reinserida na versão final, com a ficha catalográfica fornecida pela biblioteca. Informações sobre este processo devem ser obtidas na secretaria da pós-graduação.

HERON DE SOUZA MARQUES

PEIXE-DIPNOICO: UM FRAMEWORK PARA PREDIÇÃO DE MEDIDAS ESTRUTURAIS DE SOFTWARE EM SÉRIES TEMPORAIS

Dissertação apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense, como requisito parcial para obtenção do Grau de Mestre. Área de Concentração: Engenharia de Software.

Aprovada em Agosto de 2013.

BANCA EXAMINADORA

Prof. Dr. LEONARDO GRESTA PAULINO MURTA – Orientador
UFF

Prof. Dr. ALEXANDRE PLASTINO DE CARVALHO – Coorientador
UFF

Prof. Dr. JOSÉ VITERBO FILHO
UFF

Prof. Dr. TOACY CAVALCANTE DE OLIVEIRA
UFRJ

Niterói
2013

Dedico este trabalho a mim.

AGRADECIMENTOS

A Deus seja toda honra, louvor e gratidão para todo o sempre, por cumprir em minha vida a promessa de que “todas as coisas cooperam para o bem daqueles que O amam”. Agradeço-Lhe não só pelos momentos bons, que me trouxeram alegria, mas também pelos momentos difíceis, que me trouxeram aprendizado.

Agradeço à minha família, onde estão incluídos não só a que se estabelece por laços de sangue, mas também a família que àquela se agrega através dos vínculos do amor e da amizade. São meus pais Antonio e Neuza, irmão Matheus, avó Ione, primos Rafael e Melise, madrinha Leila, tias Lucia e Elza, namorada Priscilla e amigos Ricardo Nascimento Júnior e Flávia Fidelis. A todos eles agradeço o carinho de sempre e o apoio nos momentos mais trabalhosos do mestrado.

Agradeço aos meus colegas do GEMS, pelas sugestões e críticas construtivas nos *workshops* e prévias. Em especial, agradeço ao Daniel Heráclio, Gleiph Ghiotto e Daniel Castellani, principalmente pela paciência em atender aos meus complicados pedidos de ajuda.

Agradeço aos meus orientadores, Leonardo Murta e Alexandre Plastino, não só por cumprirem seu dever de orientar um aluno, mas também por apoiarem e motivarem, zelarem pela coerência da pesquisa e apresentarem valiosas contribuições ao trabalho.

Agradeço aos professores e funcionários do Instituto de Computação, em especial às secretárias Viviane e Teresa, ao técnico do suporte Carlos Eduardo e aos professores Regina Leal, Rosangela Lima, Luis Valter e Esteban Clua. Agradeço também aos professores que compuseram minha banca, professores José Viterbo Filho e Toacy Cavalcante de Oliveira.

Agradeço ao CNPQ, pelo apoio financeiro.

“A melhor maneira de predizer o futuro é criá-lo.” (Peter Drucker)

RESUMO

No contexto do desenvolvimento de software, projetos enfrentam problemas de ordem gerencial, agravados em situações em que o desenvolvimento é distribuído, heterogêneo e concorrente, isto é, quando há várias pessoas, de culturas diferentes, desenvolvendo o software ao mesmo tempo. Em geral, todo software começa com uma arquitetura bem estruturada, a qual possui características como modularidade, coesão, facilidade de manutenção, etc. No entanto, em uma equipe heterogênea, as soluções implementadas nem sempre preservam essas características à arquitetura. Para evitar a ocorrência de situações difíceis advindas da não preservação dessas características, as equipes de desenvolvimento têm à disposição uma forma de ter uma percepção a respeito do estado atual do produto em construção através de métricas. Com elas, atribuem-se valores a características do software como modularidade, coesão e facilidade de manutenção, tornando-se possível a realização de uma avaliação mais precisa acerca delas.

A percepção do estado atual do software através da avaliação de valores de métricas, como abordado previamente, evita problemas ao processo de desenvolvimento. Contudo, se a análise dos valores das métricas for feita tardiamente, isto é, muito depois do início do projeto, ele pode já estar em uma situação insustentável, sendo necessária a adoção de ações custosas ou extremas, como a refatoração ou até mesmo a sua finalização. Mesmo sendo essa análise realizada continuamente, de tempos em tempos, é possível que os valores das métricas, durante sua evolução com o passar do tempo, escondam uma tendência não facilmente detectável. Assim, conclui-se que embora a análise contínua dos valores atuais de métricas ajude a evitar a ocorrência de situações de desequilíbrio em um projeto, é importante que a evolução desses valores seja modelada, pois a observação do estado atual do software pode não revelar tendências nessa evolução. Através de técnicas de predição, essa modelagem é realizada, podendo-se assim conhecer antecipadamente os valores que as métricas em questão assumirão em momentos futuros do projeto.

Muitas abordagens foram propostas na literatura com o objetivo de realizar predições de valores de métricas de software. Essas abordagens implementam técnicas como classificação, clusterização, análise de séries temporais e redes neurais. Entretanto, atendendo a uma demanda ainda não contemplada por essas abordagens, o presente trabalho tem como objetivo apresentar um *framework* que auxilia a realização de experimentos e a construção de aplicações destinadas a realizar predições. Além do *framework*, este trabalho também apresenta os resultados de experimentos executados através desse *framework*. Esses

experimentos utilizam dados extraídos de projetos como Ant, Tomcat e Maven-3 e avaliam o desempenho de técnicas tradicionais, como Regressão Linear e Médias Móveis, e de técnicas novas, implementadas neste trabalho.

Palavras-chave: Predição, Série Temporal, Métrica, *Framework*.

ABSTRACT

In the context of software development, projects deal with managerial problems, worsen by situations in which development is distributed, heterogeneous and concurrent, that is, when there are several people, from different cultures, developing the software at the same time. Generally, every software begin with an well-structured architecture, which has characteristics such as modularity, cohesion and easiness of maintenance. However, in a heterogeneous team, implemented solutions does not always preserve these characteristics in the architecture. In order to avoid the occurrence of difficult situations that come from not preserving these characteristics, development teams rely on a way of having a perception about the recent state of the product under development through metrics. With the aid of them, values are assigned to software characteristics such as modularity, cohesion and easiness of maintenance, what makes it possible to perform a more precise evaluation about them.

The perception of the recent state of software through evaluation of metric values, as approached before, avoids problems at development process. Nevertheless, if analysis of metric values is late, that is, much time after the beginning of the project, it can already be in a unsustainable situation, being necessary to take expensive or ultimate actions, such as refactoring or even its conclusion. Even if this analysis is performed continually, it is possible that metric values, during its evolution throughout the time, hide any tendency not easily detectable. Thus, one can deduce that although continuous analysis of recent metric values can help avoid the occurrence of instability situations in a project, it is important to model evolution of these values, because looking-out recent state of software may not reveal tendencies in this evolution. Through prediction techniques, this modeling is performed, making it possible to know in advance the values that metrics will take in future stages of the project.

Several approaches have been proposed in literature aiming to perform predictions of software metric values. These approaches implement techniques as classification, clustering, time series analysis and neural networks. However, meeting a demand not covered by these approaches, this work aims to introduce a framework that aids to perform experiments and to build applications that make predictions. In addition to the framework, this work also presents the experiments' results performed by this framework. These experiments use data extracted from projects as Ant, Tomcat and Maven-3 and evaluate performance of traditional

techniques, such as Linear Regression and Moving Averages, and of new techniques, implemented in this work.

Keywords: Prediction, Time Series, Metric, Framework.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

CAPÍTULO 1 – INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO E MOTIVAÇÃO

No contexto do desenvolvimento de software, projetos enfrentam problemas de ordem gerencial. Esses problemas são agravados em situações em que o desenvolvimento é distribuído, heterogêneo e concorrente, isto é, quando há várias pessoas, de culturas diferentes, desenvolvendo o software ao mesmo tempo, visto que se torna complexo o gerenciamento de pessoas, custos, tempo e qualidade. Em geral, todo software começa com uma arquitetura bem estruturada, a qual possui características como modularidade, coesão, facilidade de manutenção, etc. No entanto, em uma equipe heterogênea, as soluções implementadas nem sempre preservam essas características à arquitetura. Dessa forma, por não haver modularidade, por exemplo, torna-se difícil realizar manutenções evolutivas ou corretivas, já que será grande a probabilidade de que uma pequena modificação em uma parte do software influencie o comportamento de outra não relacionada. Outro exemplo é a dificuldade de compreender o software devido ao baixo grau de coesão, de modo que um mesmo componente possivelmente estará modelando conceitos de entidades diferentes do domínio da aplicação. Esse tipo de dificuldade traz como principais consequências os atrasos em entregas de solicitações e o aumento de custo (devido à necessidade de alocar mais pessoas ao projeto).

Para evitar a ocorrência de cenários como aqueles descritos anteriormente, as equipes de desenvolvimento têm à disposição uma forma de ter uma percepção a respeito do estado atual do produto em construção através de métricas. Com elas, atribuem-se valores a características do software como modularidade, coesão e facilidade de manutenção, tornando-se possível a realização de uma avaliação mais precisa acerca delas. Segundo PRESSMAN (2005), as métricas auxiliam a compreender, controlar e melhorar o processo de desenvolvimento de software, e, por conseguinte, o próprio software. Assim, ao se verificar objetivamente através das métricas que, por exemplo, a modularidade ou a coesão do software está baixa, ações podem ser tomadas apropriadamente para reverter essa situação. Uma possível ação a ser tomada, baseada nos valores das métricas analisadas, é a completa reestruturação da arquitetura (refatoração), de modo que a arquitetura não mais seja a causadora dos problemas mencionados.

A percepção do estado atual do software através da avaliação de valores de métricas, como abordado previamente, evita problemas ao processo de desenvolvimento. Contudo, se a

análise dos valores das métricas for feita tardiamente, isto é, muito depois do início do projeto, ele pode já estar em uma situação insustentável, sendo necessária a adoção de ações custosas ou extremas, como a refatoração ou até mesmo a sua finalização. Mesmo sendo essa análise realizada continuamente, de tempos em tempos, é possível que os valores das métricas, durante sua evolução com o passar do tempo, apresentem uma tendência não facilmente detectável. Por exemplo, o valor da métrica relativa à complexidade do código-fonte pode estar evoluindo em tendência exponencial. Dessa forma, a análise do valor dessa métrica em um momento inicial de sua evolução não revelaria o grande problema que estaria para surgir após alguns poucos momentos e as consequências já levantadas seriam observadas posteriormente.

Assim, conclui-se que embora a análise contínua dos valores atuais de métricas ajude a evitar a ocorrência de situações de desequilíbrio em um projeto, é importante que a evolução desses valores seja modelada, pois a observação do estado atual do software pode não revelar tendências nessa evolução. Através de técnicas de predição, essa modelagem é realizada, podendo-se assim conhecer antecipadamente os valores que as métricas em questão assumirão em momentos futuros do projeto. Dessa forma, utilizando novamente o exemplo da métrica cujo valor evolui em tendência exponencial, através de uma técnica de predição que modelasse essa tendência, seria possível saber antecipadamente que a complexidade atingiria valores muito elevados, podendo-se adotar ações preventivas.

1.2 OBJETIVOS

Como pode ser visto no Capítulo 3, muitas abordagens foram propostas na literatura com o objetivo de realizar predições de valores de métricas de software. Essas abordagens implementam técnicas como classificação, clusterização, análise de séries temporais e redes neurais. Entretanto, atendendo a uma demanda ainda não contemplada por essas abordagens, o presente trabalho tem como objetivo apresentar um *framework* que auxilia a realização de experimentos e a construção de aplicações destinadas a realizar predições.

O *framework* proposto, denominado Peixe-Dipnoico[1], possui como pontos de extensão as métricas e as técnicas de predição. Assim, ele promove as pesquisas sobre o tema predição, visto que ele oferece um arcabouço para a implementação de novas técnicas e de novas

[1] O nome da abordagem, Peixe-Dipnoico, é um neologismo. Esse termo é derivado do nome de uma subclasse de peixes, a *Dipnoi*. Em matéria divulgada em meio televisivo, falou-se de uma pesquisa envolvendo um peixe dessa subclasse, na qual tem-se observado que o animal, aproximadamente quatro dias antes da ocorrência de um terremoto, aumenta a frequência dos seus movimentos no aquário. Pesquisadores investigam se o animal é capaz de realizar predições sobre a ocorrência desse fenômeno natural.

métricas. Também são pontos de extensão do *framework* o módulo de obtenção de medidas, o qual permite a utilização de diferentes fontes de dados (como repositórios dos Sistemas de Controle de Versões GIT e Subversion), e o módulo de geração de saídas, que permite a visualização dos resultados das predições e experimentos em diferentes formatos (como gráficos, histogramas e relatórios). Os dados para a predição são valores de métricas, armazenados na forma de séries temporais, que são sequências de valores ordenados de acordo com o momento de sua obtenção, como explicado no Capítulo 2.

1.3 ORGANIZAÇÃO

Essa dissertação está organizada em seis capítulos. No Capítulo 2, são apresentados os conceitos fundamentais para a compreensão do trabalho, como os conceitos predição, série temporal e métrica. No Capítulo 3, é apresentado um levantamento bibliográfico de trabalhos relacionados realizado por meio de uma Revisão Sistemática da Literatura. No Capítulo 4, a abordagem Peixe-Dipnoico é descrita e são apresentadas alguns exemplos de sua aplicação. No Capítulo 5, a abordagem é utilizada para gerenciar experimentos com o objetivo de comparar os desempenhos das técnicas implementadas através do *framework*. Por fim, no Capítulo 6 é apresentada a conclusão do trabalho, destacando-se as contribuições, limitações e trabalhos futuros.

CAPÍTULO 2 –PREDIÇÃO DE VALORES DE MÉTRICAS DE SOFTWARE POR MEIO DE SÉRIES TEMPORAIS

2.1 INTRODUÇÃO

A predição é o resultado de um processo de elaboração de uma informação antecipada sobre o acontecimento de um evento no futuro ou sobre a forma como ele ocorrerá (“Prediction”, 2013). O ato de predizer é constante no dia a dia das pessoas. Planejam-se atividades e eventos cotidianos com base em predições, mesmo que inconscientes, de situações que se espera que aconteçam. Por exemplo, o planejamento da atividade de chegar ao trabalho pode passar pela antecipação do fato de que o tráfego do trajeto estará congestionado, por se tratar do primeiro dia útil após um feriado prolongado. Em outro contexto, a decisão de ir à praia no final de semana pode ser influenciada pela observação das condições climáticas no final do dia anterior ao acontecimento do evento planejado, indicando que provavelmente irá chover.

No campo científico, a predição é alvo de estudo em diversas áreas de conhecimento e é aplicada para diversos fins. Ela é estudada em Economia (predição dos índices da Bolsa de Valores), Geologia (predição do acontecimento de eventos da natureza, como chuvas, raios e terremotos), Estatística (modelos de regressão numérica) e Ciência da Computação. Dentro da Ciência da Computação, a área de Engenharia de Software tem recebido grande atenção de pesquisadores no que se refere a predição. Predição em Engenharia de Software atua sobre aspectos do software e de seu processo de construção, como custo, tempo, falhas, qualidade e elementos relativos ao código-fonte dos programas (número de linhas de código, complexidade, etc.), a ser detalhado no Capítulo 3. Tem-se percebido nessas pesquisas que a predição no contexto do desenvolvimento de software pode reduzir custos e ajudar na diminuição de riscos de atrasos e no aumento da produtividade. Por exemplo, predizer o número de falhas na próxima etapa do processo de desenvolvimento pode ajudar o gerente de um projeto a evitar atrasos, criando uma melhor realocação de recursos humanos nas atividades de manutenção. Da mesma forma, predizer a complexidade de uma unidade de programação pode indicar a necessidade de se realizar, no presente, uma reestruturação da arquitetura do software para que seu aumento não reduza a produtividade do desenvolvimento devido a dificuldades de compreensão do código.

Considerando a definição de predição dada no início da seção, no contexto deste trabalho, a informação a ser predita é representada pelas **medidas estruturais** (Seção 2.2),

isto é, medidas que descrevem características do software quanto a seu tamanho, complexidade, elementos da linguagem de programação (número de métodos e atributos), dentre outros. O processo de elaboração de informações é representado pelas **técnicas de predição**. Essas técnicas são apresentadas no Capítulo 4 e são baseadas naquelas apresentadas na Seção 2.4. Finalmente, ainda parafraseando a definição de predição, o evento que ocorre no futuro é a variação no valor de uma das características estruturais do software. A predição assume a forma numérica, isto é, obtêm-se valores futuros, em contraste à forma categórica, na qual a informação sobre o futuro é selecionada dentre um conjunto de valores preestabelecidos. As medidas estruturais são armazenadas em conjuntos de valores denominados **séries temporais** (Seção 2.3), as quais são utilizadas como entrada para as técnicas de predição. O número de valores futuros que essas técnicas predizem, relativos à mesma métrica que descreve os valores da série de entrada, pode ser escolhido pelo usuário, sendo chamado de **horizonte de predição** (MUELLER, 1996).

Este capítulo trata dos conceitos levantados nesta seção introdutória. Na Seção 2.2, estabelece-se a diferença entre os conceitos de métrica, medida e medição, além de se apresentarem as métricas usadas no trabalho e as classificações das métricas propostas na literatura. Na Seção 2.3, o conceito de série temporal, elemento fundamental da proposta desta dissertação, é definido e são apresentados exemplos de sua utilização. A Seção 2.4 apresenta as técnicas de predição tradicionais que utilizam como insumo os valores armazenados em séries temporais. Por fim, a Seção 2.5 apresenta as considerações finais.

2.2 MÉTRICAS, MEDIDAS E MEDIÇÕES

Segundo SOLIGEN e BERHOUT (1999), a medição é a atribuição de medidas, que podem ser números ou categorias, a atributos de entidades do mundo real, de modo a descrevê-los de acordo com regras claramente definidas. Utilizando o exemplo prático de um supermercado, na pesagem de alimentos, que consiste em um processo de medição, a medida denominada *peso líquido* é atribuída à informação sobre o atributo *massa* da entidade *alimento*. A “regra claramente definida” é o algoritmo que a balança segue para determinar o valor da massa do produto, também chamada de *métrica*.

No campo de estudos da Engenharia de Software, as atividades de medição têm o papel de definir e coletar dados relativos ao desenvolvimento de um software para compreendê-lo, controlá-lo e prover informações significativas para sua melhoria. A informação gerada por esse processo é comumente chamada de *medida*, a qual é relacionada a uma determinada *métrica* de software (PRESSMAN, 2005).

Métricas de software podem ser classificadas sob vários aspectos, conforme observado na literatura (KAN, 2002; MEIRELLES, 2008). As classificações existentes não se restringem àquelas mostradas a seguir.

- Quanto ao objeto: estabelece se a métrica se refere ao produto (software), ao processo, à gestão, aos recursos, aos clientes, ao projeto, etc.
- Quanto à escala do valor: estabelece uma classificação hierárquica quanto à forma como os valores de uma métrica serão considerados. Cada categoria tem todas as características de sua antecessora, exceto aquelas que essa categoria, a sucessora, em posição mais elevada na hierarquia, modifica.
    - Nominal: os valores retornados pela métrica não possuem ordenação, são mutuamente exclusivos (cada atributo só pode ser caracterizado por uma e somente uma categoria) e cobrem todos os possíveis valores para o atributo a que se relaciona. Por exemplo, a métrica Linguagem de Programação, cujos possíveis valores são Java, C++, Ruby, etc.
    - Ordinal: os valores retornados pela métrica possuem ordenação, entretanto não é possível mensurar a magnitude das diferenças entre seus valores. Assim como a categoria Nominal, os valores são mutuamente exclusivos e cobrem todos os possíveis estados do atributo a que se relaciona. Por exemplo, a métrica Satisfação do Cliente, cujos possíveis valores são Totalmente Satisfeito, Satisfeito, Neutro, Insatisfeito e Totalmente Insatisfeito.
    - Intervalar: é possível medir a diferença numérica entre as medidas, no entanto não é possível definir um valor de origem (ponto zero). Por exemplo, as temperaturas Celsius e Fahrenheit.
    - Racional: é definido um ponto de origem (ponto zero). Por exemplo, a métrica Número de Linhas de Código, cujo ponto de origem é o valor zero.
- Quanto ao critério de avaliação: estabelece se a medida é avaliada objetivamente, sempre da mesma forma, independentemente do instante, condições ou indivíduos que as determinam (por exemplo, Número de Linhas de Código) ou se a avaliação envolve algum julgamento subjetivo (por exemplo, modelos de estimativa de custo que dependem da classificação do tipo de software).
- Quanto à composição: estabelece se a métrica é simples, obtida diretamente de uma fonte, ou composta, resultado da combinação dos valores de outras métricas.

As métricas utilizadas na abordagem desta dissertação são referentes a produto, racionais, objetivas e simples. Existem muitas métricas com essas classificações, como as

métricas da suíte de Chidamber-Kemerer (CHIDAMBER; KEMERER, 1994), que define métricas como *Depth of Inheritance*, *Coupling*, *Cohesion* e *Weighted-Method Per Class*. Além dessas, existem as métricas Número de Linhas de Código, Complexidade Ciclomática e Número de Métodos Públicos, as quais, devido à sua relevância para os experimentos, são descritas em maiores detalhes nas seções seguintes.

2.2.1 NÚMERO DE LINHAS DE CÓDIGO

Segundo KAN (2002), a métrica Número de Linhas de Código (LOC) conta o número de linhas de código existentes em um programa, porém há ambiguidades com relação à definição do que é uma linha de código. Isso acontece porque nas linguagens de programação atuais, que trabalham em alto nível de abstração, uma linha em um arquivo de código-fonte contendo uma instrução nem sempre corresponde, em linguagem de máquina, a uma única instrução, como acontecia com as linguagens de baixo nível (como Assembly). Além disso, instruções equivalentes de alta granularidade, comparadas entre duas linguagens de alto nível, podem corresponder a quantidades e tipos diferentes de instruções de baixa granularidade.

Existem diversas variações no processo de determinação do valor de LOC, dentre as quais estão:

- Contagem apenas de linhas executáveis.
- Contagem de linhas executáveis e de definição de dados.
- Contagem de linhas executáveis, de definição de dados e de comentários.

Nesta dissertação, a variação da métrica conta as linhas executáveis e de definição de dados[2]. Em outras palavras, são contados *tokens* relativos à declaração de pacote, à importação de classes (*import*), à declaração de classes, de interfaces, de campos, de métodos, de construtores, de *labels* (como *case* e *default*), à invocação de construtores e a comandos (*statements*). Não são contados comandos ou blocos vazios ou ponto-e-vírgula após fecha-parênteses. Tampouco são contados os comentários.

2.2.2 COMPLEXIDADE CICLOMÁTICA

A métrica Complexidade Ciclomática foi proposta por MCCABE (1976) para designar a facilidade de desenvolver testes (testabilidade) e facilidade de compreensão (compreensibilidade) de um programa. Baseada em teoria de grafos, essa métrica corresponde ao número de regiões nessa estrutura de dados. No software, é o número de caminhos

[2] Essa variação da métrica foi utilizada por ser a que era oferecida pela ferramenta de medição desenvolvida pelo grupo de pesquisa deste autor.

linearmente independentes que compõem o fluxo de execução de um programa. Para determinar os caminhos, o fluxo de execução do programa é representado como um grafo fortemente conectado com pontos de entrada e saída únicos. Em sua publicação, McCabe também deduziu uma fórmula (1) para calcular a Complexidade Ciclomática, na qual V(G) é a Complexidade Ciclomática de um grafo G, *e* é o número de arestas de G, *n* é o número de vértices de G e *p* é o número de partes não conectadas de G.

$$M = V(G) = e - n + 2p \quad \textbf{(1)}$$

Seja, por exemplo, o grafo apresentado na Figura 1, que representa um programa que contém apenas dois comandos de decisão. Contando arestas, vértices e partes desconectadas do grafo, percebe-se que *e* = 8, *n* = 7 e *p* = 1. Substituindo na (1, obtém-se M = 3. No trabalho de McCabe, também foi deduzido, como pode ser observado nesse exemplo, que a Complexidade Ciclomática corresponde ao número de instruções de desvio (IF, WHILE, FOR, etc.) ou booleanas (AND e OR) acrescido de uma unidade [3]. No exemplo da Figura 1, há duas instruções de desvio (representadas pelos nós *a* e *d*), o que leva, portanto, a que a complexidade seja igual a três.

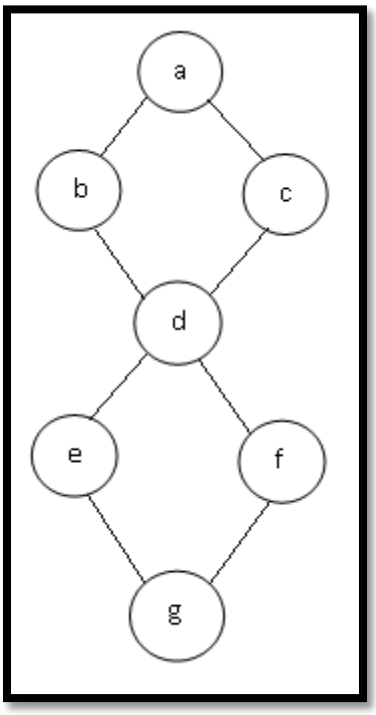

Figura 1. Grafo de um programa com dois comandos de decisão.

[3] Na publicação de McCabe, define-se que a Complexidade Ciclomática pode ser calculada com base no número de *predicates* acrescido de uma unidade.

2.2.3 NÚMERO DE MÉTODOS PÚBLICOS

A métrica Número de Métodos Públicos é a mais simples de ser entendida e medida. Constitui-se na contagem de todos os métodos que possuem o modificador de acesso *public* em sua declaração. Assim, em uma classe como a apresentada na Figura 2, o Número de Métodos Públicos seria 3.

```
public class Exemplo {
      ...
      public void metodo1()
      { ... }

      public int metodo2()
      { ... }

      public float metodo3(int parametro1)
      { ... }
      ...
}
```

Figura 2. Classe Exemplo.

2.3 SÉRIES TEMPORAIS

Uma série temporal S é um conjunto de observações $\{x_1, x_2, x_3, ..., x_N\}$, onde cada x_i ($i \in [1, \infty[$) é o registro de uma dessas observações em determinado instante *i*. A ordem de apresentação das observações determina a precedência temporal entre elas, de modo que qualquer observação x_i é mais antiga do que outra observação x_{i+t}, sendo ($t \in [1, \infty[$). Séries temporais podem ser discretas ou contínuas. Séries temporais discretas contêm observações obtidas em instantes de tempo definidos, equidistantes ou não. Nesse tipo de série, existem intervalos de tempo entre os valores em que não se realizou a medição, por livre escolha ou por inviabilidade. São exemplos desse tipo de série aquelas que representam o número de vendas de carros a cada mês em uma concessionária, ou que representam o número de assaltos a cada mês em uma cidade. Quando as observações podem ser obtidas continuamente ao longo de um intervalo de tempo, diz-se que a série temporal por elas formada é contínua. Essencialmente, a série contínua pode ser considerada uma série discreta cujos intervalos de tempo são os menores com que o instrumento de medição utilizado consegue lidar. São exemplos desse tipo de série aquelas que representam a velocidade instantânea de um automóvel em um trajeto (BROCKWELL; DAVIS, 2002).

Como discutido anteriormente, séries temporais são utilizadas em vários campos do conhecimento como Economia (preços diários de ações, taxa mensal de desemprego, produção industrial), Medicina (eletrocardiograma, eletroencefalograma), Epidemiologia

(número mensal de novos casos de meningite), Meteorologia (precipitação pluviométrica, temperatura diária, velocidade do vento), etc. Independentemente do campo de conhecimento, geralmente os principais objetivos em se estudar séries temporais são os seguintes (EHLERS, 2009):

- Descrição. Descrever propriedades da série como o padrão de tendência, observações discrepantes (*outliers*), alterações estruturais (por exemplo, mudanças no padrão de tendência de crescimento), etc.
- Explicação. Usar a variação em uma série para explicar a variação em outra série.
- Predição. Predizer valores futuros com base em valores passados. Nesse caso, assume-se que o futuro envolve incerteza, ou seja, as previsões não são perfeitas.
- Controle. Controlar a qualidade de um processo a partir da análise dos dados da série.

Nesta dissertação, uma série temporal $S = \{x_1, x_2, x_3, ..., x_N\}$ é um conjunto de medidas estruturais extraídas das várias versões do código-fonte de um software. Nesse conjunto, as medidas são sempre ordenadas da mais antiga para a mais recente. As séries temporais são discretas, já que são obtidas de revisões armazenadas em um Sistema de Controle de Versões.

2.4 TÉCNICAS DE PREDIÇÃO UTILIZANDO SÉRIES TEMPORAIS

Como dito na seção 2.3, um dos objetivos do estudo de séries temporais é encontrar uma forma eficiente de utilizá-las para a predição. Através de técnicas de predição, inter-relacionamentos entre características de observações passadas, contidos nessas séries, integram modelos que geram predições a partir da extrapolação dessas características. Segundo MUELLER (1996), a maioria das técnicas tem a premissa de que as observações passadas contêm todas as informações necessárias sobre o padrão de comportamento da série temporal, sendo esse padrão recorrente no tempo.

Embora quase todas as técnicas de predição analisem apenas as observações armazenadas em uma série de interesse (entrada), algumas utilizam o comportamento de outras séries para modelar o comportamento dessa série de interesse. Dessa forma, as técnicas são classificadas como *univariadas*, *funções de transferência* e *multivariadas* (MUELLER, 1996). As técnicas univariadas consideram somente uma única série (a entrada) para a realização das predições. Já funções de transferência é uma categoria de técnicas que utilizam

não somente seus dados históricos, mas também os dados de outras séries temporais não correlatas entre si, com a ressalva de que seja conhecida a relação de causalidade entre essas séries. Por fim, as técnicas multivariadas consideram mais de uma série temporal para a realização de predição, tendo elas relação de causalidade ou não entre si.

Há uma grande variedade de técnicas simples e avançadas de predição de séries temporais (MORETTIN; TOLOI, 1981; MUELLER, 1996; BROCKWELL; DAVIS, 2002; EHLERS, 2009). Algumas delas, por terem maior relevância no contexto deste trabalho, são detalhadas nas subseções seguintes. Essas técnicas são usadas em áreas de conhecimento como Economia e Engenharia. São as técnicas *Médias Móveis*, *Regressão Linear* e *Regressão Quadrática*, todas elas univariadas. Também é detalhada uma técnica multivariada, a *k-NN*, baseada em uma técnica homônima da área de aprendizagem de máquina, que figura neste capítulo também devido à sua relevância ao contexto deste trabalho.

2.4.1 MÉDIAS MÓVEIS

A técnica de Médias Móveis compõe uma predição com a média aritmética das observações passadas recentes. Segundo MORETTIN (1981), a predição para o primeiro instante de tempo t após a última observação registrada, utilizando a Médias Móveis, é dada por (**2**), onde n é o número de observações incluídas na média (também chamado de **janela**), y_{t+1} é o valor predito e y_1, y_2, ..., y_t são os valores registrados na série temporal.

$$y_{t+1} = \frac{y_t + y_{t-1} + ... + y_{t-n}}{n} \quad \textbf{(2)}$$

Utiliza-se o termo *média móvel* porque a média das observações é recalculada à medida que a próxima observação se torna disponível. A observação nova é incluída na janela, em detrimento da observação mais antiga. Dessa forma, a inclusão dos dados para o cálculo da média funciona como se a janela fosse deslocada à direita para englobar a nova observação, como é exemplificado na Figura 3, na Figura 4 e na Figura 5. Em cada figura, o tamanho da janela, representada pelo retângulo preto, é 2. Os pontos interligados formam uma série de entrada e os pontos desconexos são os novos valores observados. Na Figura 3, as últimas duas observações, marcadas pelo retângulo preto, compõem uma operação de obtenção de média. Na Figura 4, a janela se move para a direita e engloba o primeiro novo valor, o qual é utilizado com o último da entrada para gerar a nova predição. Já na Figura 5, os dois novos valores compõem a predição.

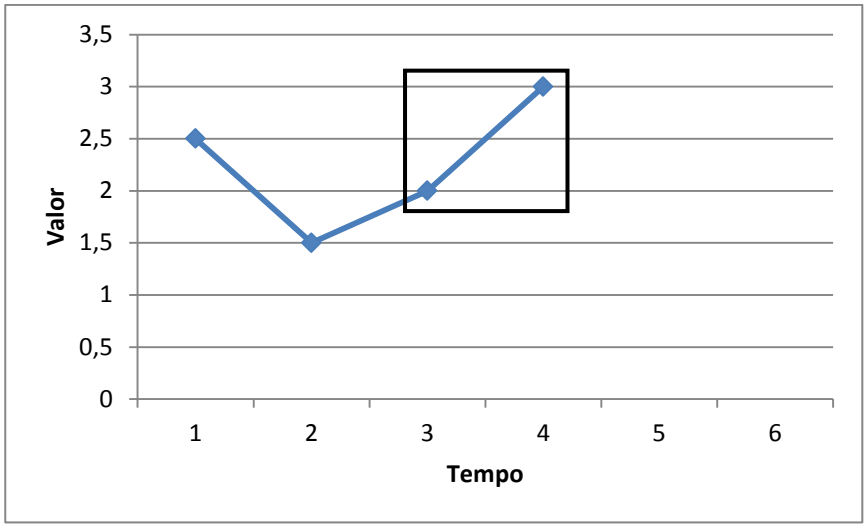

Figura 3. Janela em movimento para o cálculo das médias: Instante 1.

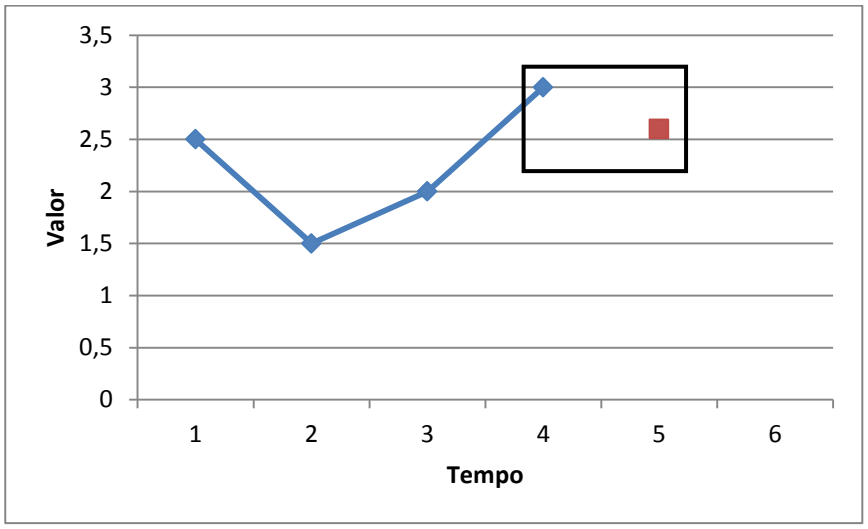

Figura 4. Janela em movimento para o cálculo das médias: Instante 2.

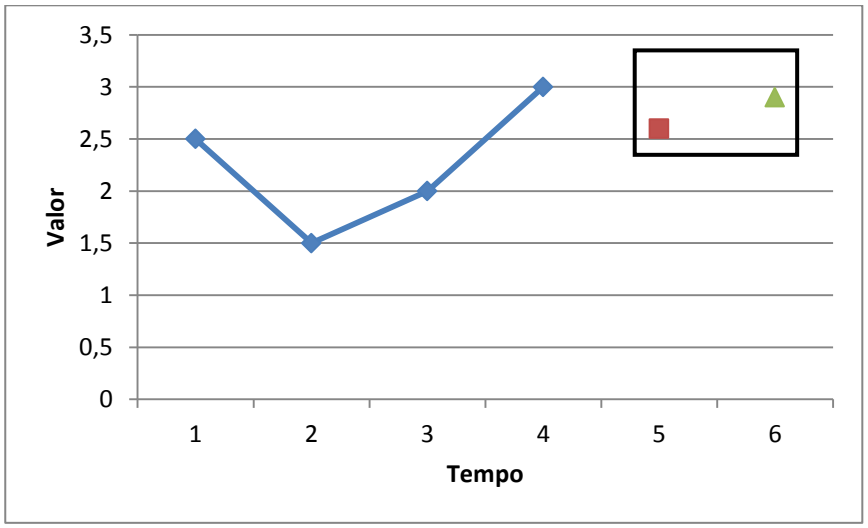

Figura 5. Janela em movimento para o cálculo das médias: Instante 3.

2.4.2 AJUSTE DE FUNÇÕES

Segundo CLÁUDIO E MARINS (1994), técnicas de ajuste de funções são técnicas de aproximação que, aplicadas a um conjunto de dados experimentais $\{(x_1,y_1), (x_2,y_2), ..., (x_N,y_N)\}$, obtêm uma função y = f(x) que relaciona x com y de modo a gerar o menor erro. A partir daí, torna-se possível calcular o valor de y relativo a um valor de x que não conste nos dados experimentais. Realizando o ajuste de funções em um conjunto de dados experimentais armazenados em uma série temporal, torna-se possível obter informações de instantes de tempo (variável independente) que estão além do último registrado, ou seja, predições.

Para encontrar a função que melhor se ajuste aos dados experimentais (isto é, que gere o menor erro), utiliza-se a técnica de Aproximação por Mínimos Quadrados (CLÁUDIO; MARINS, 1994). Sendo o erro R_i de cada aproximação f_i^* com relação ao valor real y_i igual a $f_i^* - y_i$, a Aproximação por Mínimos Quadrados tem o objetivo de minimizar $\sum_{i=1}^{N} R_i^2$. A partir dessa técnica, é deduzido um sistema de equações, que é utilizado para a obtenção de um conjunto de constantes (coeficientes ou parâmetros) c_j de uma função f^*. Essa função aproxima uma função desconhecida f (que gerou os dados experimentais) e possui a forma mostrada em **(3)**. Nessa equação, as constantes c_j, $1 \leq j \leq N$, são lineares.

$$f^*(x) = c_1\phi_1(x) + c_2\phi_2(x) + ... + c_N\phi_N(x) \quad \textbf{(3)}$$

As funções ϕ_1, ϕ_2, ..., ϕ_N são escolhidas de acordo com a natureza dos dados experimentais, ou seja, se os dados se comportam como se fosse regidos por uma função linear, a função ϕ_1 é igual a x e ϕ_2 é uma função constante qualquer; se os dados assumem o comportamento de uma parábola, ϕ_1 é x^2, ϕ_2 é x e ϕ_3 é uma função constante qualquer; e assim por diante. Ao final da dedução matemática, chega-se ao sistema de equações mostrado na Figura 6, onde $A_{jk} = \sum_i \phi_j(x_i)\phi_k(x_i)$, $1 \leq j \leq n$, $1 \leq k \leq n$, e n é o número de termos da função f^*.

Para exemplificar, a Tabela 1 contém pares ordenados (x_i,y_i) de dados experimentais.

x_i	10,0	10,1	10,2	10,3	10,4	10,5
f_i	1	1,20	1,25	1,267	1,268	1,274

Tabela 1. Dados experimentais.

Observando-se que os dados podem ser ajustados adequadamente a uma reta, cuja expressão é $f^*(x) = a_0 + a_1x$, segue-se que $\phi_1(x) = 1$ e $\phi_2(x) = x$. Assim, para n = 2, obtêm-se os coeficientes A_{jk} do sistema de equações a ser resolvido (Figura 6), como mostra a Figura 7.

$$A_{11}c_1 + A_{12}c_2 + \cdots + A_{1n}c_n = \sum_i \phi_1(x_i)f_i$$
$$A_{21}c_1 + A_{22}c_2 + \cdots + A_{2n}c_n = \sum_i \phi_2(x_i)f_i$$
$$\vdots$$
$$A_{n1}c_1 + A_{n2}c_2 + \cdots + A_{nn}c_n = \sum_i \phi_n(x_i)f_i$$

Figura 6. Sistema de equações para determinação dos coeficientes de f^*.

$A_{11} = \sum_i \phi_1(x_i)\phi_1(x_i) = \sum_i 1 = 6$
$A_{12} = A_{21} = \sum_i \phi_1(x_i)\phi_2(x_i) = \sum_i x = 61{,}5$
$A_{22} = \sum_i \phi_2(x_i)\phi_2(x_i) = \sum_i x^2 = 630{,}55$
$\sum_i \phi_1(x_i)f_i = \sum_i 1 \times f_i = 7{,}259$
$\sum_i \phi_2(x_i)f_i = \sum_i x_i \times f_i = 74{,}4843$

Figura 7. Coeficientes do sistema de equações.

O sistema de equações resultante é mostrado na Figura 8.

$$\begin{cases} 6a_0 + 61{,}5a_1 = 7{,}259 \\ 6a_0 + 630{,}55a_1 = 74{,}4843 \end{cases}$$

Figura 8. Sistema de equações.

A solução desse sistema é $a_0 = -3{,}449$ e $a_1 = 0{,}454$. Assim, a função f^* que oferece o melhor ajuste aos dados experimentais é $f^*(x) = -3{,}449 + 0{,}454x$. Para obter o valor da função para o instante futuro $x_i = 10{,}6$, basta utilizar esse valor como parâmetro da expressão, o que levará ao valor 1,3634.

O sistema de equações parametrizável da Figura 6 pode ser utilizado para qualquer função cujos coeficientes apareçam linearmente, ou seja, as funções ϕ_1, ϕ_2, ..., ϕ_N podem ser de qualquer forma, como $\phi_1(x) = e$, $\phi_2(x) = \text{sen}(x)$, etc., mas não podem ser do tipo $\phi_1(x) = ae^{bx}$, $\phi_2(x) = \text{sen}(ax)$ ou outras, onde os coeficientes desconhecidos não apareçam linearmente. Assim, abre-se a possibilidade de, através desse *framework* matemático, desenvolver outras funções de aproximação, como a Regressão Linear (aproximação por reta) e a Regressão Quadrática (aproximação por parábola). Podem-se desenvolver também aproximações por

polinômios de graus acima de 3, funções exponenciais, logarítmicas, dentre muitas outras opções.

2.4.3 K-NN

As técnicas da área de aprendizagem de máquina têm o objetivo de simular o processo de aprendizagem e, consequentemente, adquirir conhecimento automaticamente. Geralmente, essas técnicas utilizam experiências anteriores para auxiliar no processo de obtenção de conhecimento e melhorar seu desempenho (FERRERO, 2009).

Em técnicas denominadas **supervisionadas**, essa experiência é representada por um conjunto E de N exemplos (ou instâncias) de treinamento $E = \{(x_1,y_1), (x_2,y_2), ..., (x_N, y_N)\}$. Os valores y_i são classes (ou categorias) que rotulam um vetor de valores discretos ou contínuos $(x_{i1},x_{i2}, ..., x_{iN})$ relacionados a um conjunto de atributos $X = \{A_1, A_2, ..., A_N)$. Em outras palavras, x_{il} denota o valor do atributo A_l do exemplo *i*. A partir desse conjunto de instâncias de treinamento, a técnica constrói um modelo que é capaz de mapear um novo vetor de valores de atributos (isto é, uma nova instância) não classificado a uma das classes que constem nas instâncias desse conjunto, realizando, assim, a predição da classe desconhecida da nova instância. Para valores nominais das classes, a técnica é denominada **classificação**, enquanto que para valores numérico, a técnica é denominada **regressão** (MAIMON; ROKACH, 2010).

A técnica supervisionada conhecida como *k-NN*, sigla para *k-Nearest Neighbours*, prediz o valor da classe de uma nova instância comparando-a com outras instâncias cujas classes sejam conhecidas e atribuindo a ela a classe predominante das *k* instâncias mais parecidas (FERRERO, 2009). A Figura 9 mostra um conjunto de instâncias que podem assumir duas possíveis classes: positivo e negativo. Na execução do *k-NN*, procurando $k = 1$ vizinho mais parecido, uma nova instância E_i seria rotulada com a mesma classe do seu vizinho mais próximo, que é positivo.

Para o seu funcionamento, o *k-NN* necessita da definição de três variáveis, que são:

1. Conjunto de instâncias de treinamento, com as quais a instância nova será comparada.
2. Uma métrica que quantifique a similaridade entre a instância nova e cada instância de treinamento. Em geral, a similaridade é calculada com base em uma métrica de distância. Considerando os atributos das instâncias como pontos no espaço multidimensional, existe uma métrica genérica, denominada Métrica de Minkowsky (TAN, 2006), para calcular a distância entre dois

pontos no espaço n-dimensional de acordo com o parâmetro *d*, como é mostrado em **(4)** (FERRERO, 2009), sendo E_i e E_j instâncias. Nessa equação, quando *d* = 2, tem-se a Distância Euclideana, uma das mais comumente utilizadas. A partir de estudo de AGGARWAL et al. (2001), observou-se que *d* menores produzem os melhores resultados.

3. O valor de *k*, que indica o número de vizinhos mais próximos que devem ser considerados.

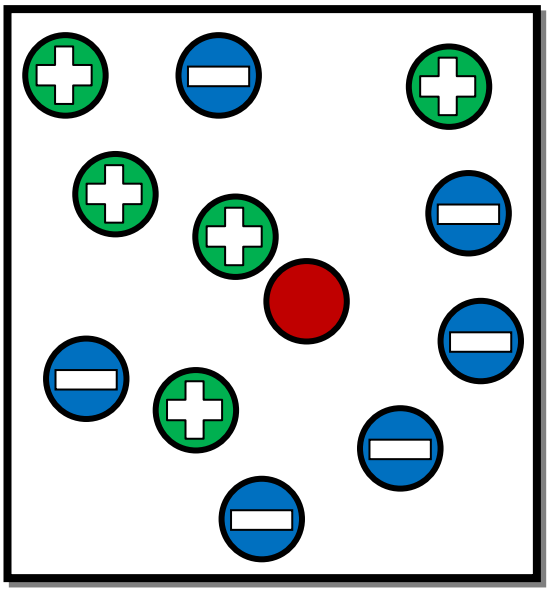

Figura 9. Exemplo de funcionamento do k-NN para a classificação de uma nova instância.

$$dist(E_i, E_j) = \sqrt[d]{\sum_{k=1}^{n} | x_{ik} - x_{jk} |^d} \quad \textbf{(4)}$$

Em FERRERO (2009), a abordagem tradicional do *k-NN* foi adaptada para realizar a predição de valores contínuos utilizando como entrada uma série temporal. A ideia em que se baseia essa adaptação é, considerando os últimos *w* valores da série (*input*), encontrar subsequências, na própria série ou em outras, de tamanho *w* que apresentem comportamentos similares. Com base nos valores contidos nessas subsequências, realiza-se a predição do provável próximo valor a constar na série. Para sua execução, é necessário determinar:

1. Tamanho *w* da janela para extrair as subsequências. Esse é o tamanho das subsequências extraídas da série temporal.
2. Conjunto de séries de treinamento. Refere-se às subsequências que serão consideradas para constituir o conjunto de séries de treinamento.
3. Métrica da similaridade. Quantifica a similaridade entre as subséries. Cada subsequência é representada no espaço *w*-dimensional, de modo que cada um

de seus pontos é considerado como o valor de um atributo no cálculo da distância (no *k-NN* tradicional). Além de calcular a similaridade através de uma métrica de distância, também é levada em consideração a distância temporal (em termos de ser mais antigo ou mais recente) entre a série de entrada e as subsequências do conjunto de séries de treinamento.

4. Cardinalidade do conjunto de séries similares. Define o número de (sub)séries mais próximas que serão consideradas para a composição do valor futuro, isto é, o valor de *k*.
5. Função de predição. Determina como serão utilizados os valores das subsequências mais próximas para estimar o valor futuro. Essa estimativa pode ser realizada, por exemplo, através da média dos valores subsequentes dessas subsequências (considerando que, no contexto delas, são seus valores futuros).

A Figura 10 ilustra as variáveis apresentadas no parágrafo anterior. No exemplo da figura, o tamanho da janela *w* é 4, correspondendo aos quatro últimos elementos da série (pontos vermelhos). O conjunto de séries de treinamento é composto por subsequências, também de tamanho *w*, obtidas da mesma série de origem dos elementos contidos na janela. Desse conjunto, são escolhidas as $k = 3$ subsequências (pontos verdes) que sejam mais similares ao *input*, de acordo com a métrica de similaridade. Obtendo os valores subsequentes às subsequências mais similares, calcula-se com eles o dado previsto (quadrado vermelho) através de uma função de previsão (por exemplo, a média aritmética).

2.5 CONSIDERAÇÕES FINAIS

A predição é uma atividade constante no dia a dia das pessoas. Além de cotidiana, é alvo de estudo em diversas áreas de conhecimento, inclusive na Engenharia de Software. O Capítulo 3 mostra trabalhos relacionados à predição em software. Seus objetivos são a redução de custos e diminuição dos riscos através da predição de informações como o número de falhas na próxima etapa do processo de desenvolvimento e a complexidade de uma unidade de programação.

Existem muitas abordagens para a realização de predições. Muitas abordagens utilizam como base para a predição as relações de causalidade contidas em séries temporais para modelar o comportamento das observações medidas. Obtido o modelo de comportamento, essas abordagens conseguem estimar os valores que virão em tempos posteriores ao último em que houve medição. São exemplos dessas abordagens a *Médias Móveis*, *Ajuste de Funções* e *k-NN*, descritas na Seção 2.4.

No Capítulo 3, como já mencionado, serão apresentados trabalhos relacionados à predição de características estruturais do software. Essas características são representadas pelas métricas estruturais, como as que foram descritas na Seção 2.2. Utilizando diversas técnicas, como Regressão Linear, técnicas de clusterização e de classificação e também Análise de Séries Temporais, esses trabalhos realizam predições relativas a falhas, custo e qualidade, dentre outros aspectos.

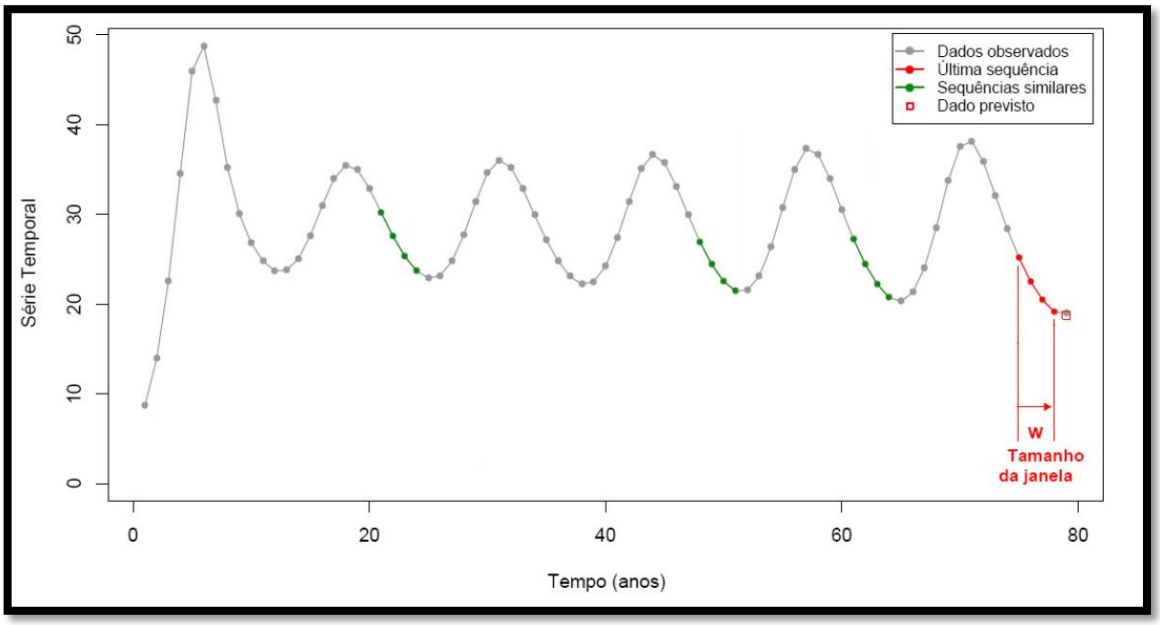

Figura 10. Ilustração das variáveis necessárias à execução do *k-NN* (FERRERO, 2009).

CAPÍTULO 3 – PREDIÇÃO DE CARACTERÍSTICAS ESTRUTURAIS DE SOFTWARE

3.1 INTRODUÇÃO

O capítulo de trabalhos relacionados desta dissertação foi elaborado sob os moldes de um estudo baseado em Revisão Sistemática da Literatura (MONTONI, 2007) com o objetivo de identificar abordagens que realizem predição de características estruturais de software. Com esse estudo, pretende-se que a revisão da literatura realizada nesta dissertação (considerando o tema central das contribuições deste trabalho) esteja menos exposta aos riscos das revisões informais, conforme descrito por BARRETO (2011). Para a sua realização, foi executado o processo de apoio à condução de estudos baseados em revisão sistemática definido por MONTONI (2007). A estrutura do texto desta seção foi moldada de forma similar ao trabalho de BARRETO (2011).

Este capítulo está estruturado da seguinte forma. Na Seção 3.2, são apresentados os passos adotados para o planejamento e organização do processo de pesquisa. Na Seção 3.3, foi proposto um processo de teste e refinamento para se chegar à expressão de busca utilizada nas bibliotecas digitais de onde foram obtidas publicações sobre predição de características estruturais de software. A Seção 3.4 descreve os procedimentos de seleção de artigos obtidos das bibliotecas digitais. A Seção 3.5 descreve os procedimentos seguidos para a extração de conhecimento das publicações obtidas. A Seção 3.6 e a Seção 3.7 apresentam, respectivamente, análises qualitativas e quantitativas das publicações, baseadas no agrupamento de abordagens com características semelhantes quanto à técnica utilizada, propósito, fonte de dados para a predição, dentre outras. Por fim, a Seção 3.8 traz as considerações finais.

3.2 DEFINIÇÃO DO PROTOCOLO DE PESQUISA

A definição de um protocolo de pesquisa é, em síntese, uma etapa de planejamento e organização do processo de pesquisa. Ela ajuda a aumentar a objetividade da atividade, guiando o pesquisador a definir o escopo, a fonte de referências e a procurar elementos nas referências que respondam concisamente à sua Questão de Pesquisa.

Assim, para este estudo baseado em revisão sistemática da literatura, foi definido o seguinte objetivo: *examinar publicações científicas que apresentem abordagens (técnicas, algoritmos, metodologias, ferramentas) que analisem medidas estruturais de um projeto de*

software, fornecendo, a partir da análise de sua evolução ao longo do tempo, predições sobre seus valores futuros.

A fim de obter as informações desejadas, este estudo baseado em revisão sistemática possui uma questão principal de pesquisa (QP) e algumas questões secundárias (QS) definidas a seguir.

QP: Como as abordagens existentes na literatura utilizam medidas estruturais de software para construir predições?

QS1: De onde são obtidos os dados para a predição?

QS2: Qual a granularidade dos artefatos de software analisados para construir a predição?

QS3: Que tipos de medidas estruturais são usados?

QS4: Para que propósito a abordagem é destinada?

QS5: Qual a técnica utilizada para construir a predição?

QS6: Quais métricas de desempenho foram utilizadas para avaliar a abordagem descrita?

QS7: Qual o desempenho aproximado das predições fornecidas pela abordagem?

QS8: Foi especificada a existência de apoio ferramental? Se sim, qual?

A fim de definir o escopo deste estudo, foram estabelecidos alguns critérios para garantir de forma equilibrada a viabilidade da execução (custo, esforço e tempo), acessibilidade aos dados e abrangência do estudo. A pesquisa deveria ser realizada a partir de bibliotecas digitais através dos seus respectivos mecanismos de busca.

Para seleção das bibliotecas digitais, foram adotados alguns critérios. Para serem consideradas, as bibliotecas digitais deveriam:

- Possuir mecanismo de busca que permitisse o uso de expressões lógicas ou mecanismo equivalente;
- Possuir mecanismo que permitisse a busca no texto completo ou em campos específicos das publicações;
- Garantir resultados únicos através da busca de um mesmo conjunto de palavras-chave no mesmo momento de tempo;
- Pertencer a uma das editoras listadas no Portal de Periódicos da CAPES; e,
- Incluir em sua base publicações da área de exatas ou correlatas que possuíssem relação direta com o tema a ser pesquisado.

Decidiu-se considerar publicações em língua inglesa, pelo fato de a grande maioria dos periódicos e conferências internacionais ter esse idioma definido como padrão. As bibliotecas digitais selecionadas foram:

- Scopus (em modo Advanced Search): <http://www.scopus.com>
- IeeeXplore (IEEE) (em modo Command Search): <http://ieeexplore.ieee.org>

3.3 MÉTODO DE BUSCA E TESTE DE PROTOCOLO

Para se chegar à expressão de busca utilizada nas bibliotecas digitais, foi realizado um processo de teste e refinamento. O primeiro passo foi determinar os parâmetros de população, intervenção, comparação e saída.

(P) População (*Population*): Pesquisas relacionadas a abordagens que utilizam medidas estruturais de software.

(I) Intervenção (*Intevention*): Técnicas de predição.

(C) Comparação (*Comparison*): Não há, pois não se tem como objetivo comparar abordagens, e sim, caracterizá-las.

(O) Saída (*Output*): Caracterizações, Abordagens, Métodos, Metodologias, Procedimentos, Mecanismos, Experiências, Descobertas, Pesquisas, Estudos, Técnicas, Conhecimentos, Ferramentas ou Resultados.

O segundo passo foi determinar as expressões que instanciassem a população, a intervenção e a saída. Dessa forma, após alguns testes para determinar se as expressões eram abrangentes o suficiente, assim foram definidas:

(P): ("structural metric*" OR "structural characteristic*" OR "structural measure*" OR "complexity measure*" OR "complexity metric*" OR "code measure*" OR ("source code" AND measure*) OR "code metric*" OR ("source code" AND metric*) OR "code attribute*" OR ("source code" AND attribute*))

(I): (predict* OR forecast)

(O): (characterization OR approach* OR method* OR methodolog* OR procedure* OR mechanism* OR experience* OR finding* OR research* OR stud* OR technique* OR knowledge OR tool* OR result*)

O terceiro passo foi determinar quais artigos seriam considerados como controle e aperfeiçoamento da expressão de busca. Esses artigos deveriam ser retornados pela busca. Caso não estivessem presentes, a expressão de busca deveria ser refinada até que todos os artigos de controle fossem retornados. Assim, pela semelhança com o tema de interesse, os seguintes artigos foram considerados relevantes para a pesquisa:

- ZHOU, Y.; XU, B.; LEUNG, H. On the ability of complexity metrics to predict fault-prone classes in object-oriented systems. *Journal of Systems and Software*, v. 83, p. 660–674, abr. 2010.
- CAMARGO CRUZ, A. E. Exploratory study of a UML metric for fault prediction. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, ICSE'10, Cape Town, South Africa: ACM, 2010. 361–364 p. v. 2.
- SHIN, Y.; BELL, R.; OSTRAND, T.; WEYUKER, E. Does calling structure information improve the accuracy of fault prediction? In: INTERNATIONAL WORKING CONFERENCE ON MINING SOFTWARE REPOSITORIES, Vancouver, Canada: IEEE Computer Society Washington, 16 maio 2009. 61–70 p.
- NAGAPPAN, N.; BALL, T.; ZELLER, A. Mining metrics to predict component failures. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, Shanghai, China: ACM Press, 2006. 452–461 p.
- KNAB, P.; PINZGER, M.; BERNSTEIN, A. Predicting defect densities in source code files with decision tree learners. In: INTERNATIONAL WORKSHOP ON MINING SOFTWARE REPOSITORIES, Shanghai, China: ACM Press, 2006. 119–125 p.

Ao final do processo descrito nesta seção, as expressões de busca foram elaboradas. Independentemente das diferenças de sintaxe existentes, as expressões foram compostas pela conjunção dos parâmetros de população, intervenção e saída, definidos anteriormente.

3.4 PROCEDIMENTOS DE SELEÇÃO E CRITÉRIOS

Os procedimentos de seleção dos artigos foram realizados em três etapas:

1ª Etapa – Seleção e Catalogação Preliminar. Esta etapa foi realizada a partir da utilização da expressão de busca nas bibliotecas selecionadas. Todas as publicações retornadas foram catalogadas para análise posterior e aplicação dos filtros de exclusão.

2ª Etapa – Seleção das Publicações Relevantes (1º Filtro). A seleção das publicações através dos critérios de busca não garante que todas as publicações selecionadas sejam úteis no contexto da pesquisa, pois a aplicação dos critérios de busca é restrita ao aspecto sintático. Assim sendo, após a identificação das publicações preliminares, os resumos (abstracts) foram lidos e classificados de acordo com os Critérios de Exclusão (CE) a seguir:

- CE1: Publicações que não propõem técnicas (metodologias, algoritmos, ferramentas) de predição baseadas em medidas estruturais de software.
- CE2: Publicações que descrevem e/ou apresentam *keynote speeches*, tutoriais, cursos, *workshops*, comentários sobre publicações, descrições de *proceedings*, e similares.
- CE3: Publicações não disponíveis para *download* em sua forma completa nas bibliotecas digitais.
- CE4: Publicações que, mesmo tendo sido retornadas a partir da expressão de busca, não estão escritas em língua inglesa.

Cada publicação foi selecionada para a próxima etapa somente se não fosse enquadrada em algum dos critérios de exclusão descritos. Para diminuir o risco de que uma publicação fosse excluída prematuramente, em caso de dúvida a publicação não foi excluída.

3ª Etapa – Seleção das Publicações Relevantes (2º Filtro). Apesar de limitar o universo de busca, o filtro aplicado na 2ª etapa não garante que todo o material coletado seja útil no contexto da pesquisa, uma vez que a seleção das publicações considerou a análise apenas do resumo/abstract da publicação. Assim, as publicações selecionadas na 2ª etapa foram lidas e filtradas de acordo com os mesmos critérios de exclusão definidos na etapa anterior.

3.5 PROCEDIMENTOS PARA EXTRAÇÃO E ARMAZENAMENTO DOS DADOS

Para cada publicação retornada pela expressão de busca (1ª etapa do procedimento para seleção das publicações), todos os dados disponíveis da publicação foram armazenados.

Cada publicação armazenada foi examinada e submetida aos filtros das duas etapas seguintes. As publicações eliminadas na 2ª ou 3ª etapa foram identificadas com “CE[número do critério de exclusão não atendido]”. As que não foram eliminadas foram identificadas com "OK". A lista das publicações, com as indicações de exclusão ou de aceitação (isto é, marcações de “OK”), encontra-se no Apêndice B.

Para cada publicação selecionada na 3ª etapa, foram registrados os dados completos da referência, um resumo, observações e a resposta a cada uma das questões secundárias, considerando a publicação. Todas essas informações obtidas das 51 publicações selecionadas na 3ª etapa estão no Apêndice A.

3.6 ANÁLISE QUALITATIVA

A partir dos registros das informações obtidas das publicações selecionadas na 3ª etapa, percebe-se que os artigos analisados trazem abordagens com visões diferentes sobre predição e utilizam técnicas diversas para alcançar propósitos variados. Assim, nesta seção diferenças e pontos em comum serão levantados.

Muitos dos trabalhos encontrados propuseram uma técnica nova, diferente das tradicionais tarefas de Classificação e de Regressão e das técnicas como Redes Neurais, por exemplo, dentre outras. Eles utilizam fórmulas e metodologias exclusivas para atingir o propósito de predição desejado. Em CAPILUPPI *et al.* (2007), é apresentado um modelo para predição de esforço antirregressivo (esforço necessário para reduzir a complexidade do software). A partir de medidas estruturais e do número de *releases* em que houve mudanças, é construída uma lista em ordem decrescente com recomendações das funções às quais se deverá dar maior atenção durante o desenvolvimento da próxima *release*. SHARAFAT *et al.* (2007) apresentam uma abordagem probabilística para predição da ocorrência de modificações na próxima *release* de uma classe. É utilizada uma fórmula composta pela Probabilidade de Mudança Interna (probabilidade de uma classe ser modificada devido a mudanças originadas em sua estrutura) e pela Probabilidade de Propagação (probabilidade de uma mudança se propagar de uma classe a outra) cujo resultado é a probabilidade total de mudança para cada classe. Com esse resultado, prediz-se se determinada classe será modificada ou não. TAKAHASHI *et al.* (1989) propõem uma fórmula para a predição do tamanho do programa, formada por métricas de Halstead extraídas de diagramas de lógica do programa escritos em notação HCP. Em SHAW W. H. *et al.* (1989), é apresentada uma abordagem para a predição do tempo de compilação para compiladores Ada. O modelo de predição é uma fórmula composta por métricas de Halstead. Em PIGHIN *et al.* (2005), uma medida de risco $R_{j,k}$ de um módulo j em sua k-ésima *release*, na versão corrente do projeto, é calculada como o produto entre sua Complexidade Ciclomática de McCabe (C_j) e o seu Fator de Persistência ($F_{j,k}$), relacionado à idade desse módulo. Os módulos são particionados em duas classes: classe Alto Risco, se seu $R_{j,k}$ for maior que um determinado valor (*threshold*), ou Baixo Risco, se seu $R_{j,k}$ for menor ou igual a esse mesmo valor. Em SHIN *et al.* (2012) e em SHIN *et al.* (2009), propõe-se um modelo para predição de falhas que utiliza medidas estruturais denominadas “Calling Structure” (também proposta no artigo) combinadas com métricas de código comumente usadas. Esse modelo constitui-se de uma metodologia que utiliza a técnica de regressão *Negative Binomial Regression*.

Duas abordagens utilizam a técnica *Bayesian Belief Network (BBN)*. Em HADERER *et al.* (2010), é apresentado o *framework* Squaner, o qual, além de detectar padrões, antipadrões e anomalias de um projeto e realizar avaliações sobre a qualidade do software, também realiza predições de falhas, utilizando BBN, na forma de probabilidades de ocorrência de falhas nos próximos seis meses. Em WANG *et al.* (2002), uma BBN foi utilizada com a intenção de predizer a data apropriada para a *release* de uma nova versão de um produto, de modo que ele possua qualidade equivalente à de versões anteriores de produtos semelhantes.

TONU *et al.* (2006) não especificam a técnica de predição utilizada. O artigo apresenta um *framework* baseado em métricas para realizar a avaliação da estabilidade de sistemas. A abordagem é conduzida em três etapas: (1) Extração da arquitetura do sistema-alvo a partir da análise de seu código-fonte; (2) Análise retrospectiva da evolução da estabilidade do sistema ao longo do tempo; (3) Determinação (predição) de um conjunto de mudanças prováveis e de seu impacto ao serem aplicadas.

A técnica de Análise de Séries Temporais foi utilizada em KHOSHGOFTAAR *et al.* (1995). O artigo mostra uma abordagem para a predição do número de falhas da próxima construção de um projeto. Foi utilizado o modelo de predição ARIMA (Autoregressive Integrated Moving Averages), no qual se assume que predições são baseadas no histórico, armazenado em um série temporal (Seção 2.3), de observações (medidas) passadas.

Foram encontradas técnicas compostas, as quais escolhem, dentre algumas opções, modelos de predição que apresentem, em uma fase de treinamento, o melhor desempenho (técnicas dinâmicas) ou combinam as funcionalidades dos modelos de predição disponíveis (técnicas híbridas). LI *et al.* (2011) mostram um método (referido como robusto) que seleciona, em uma etapa inicial, de um conjunto predefinido, algoritmos de predição que se mostraram mais qualificados e os integra, compondo um modelo de predição híbrido. PIZZI *et al.* (2006) apresentam uma abordagem para a predição do nível de complexidade e manutenibilidade que utiliza a técnica *Fuzzy Aggregation*, a qual possibilita a agregação de vários algoritmos de predição, escolhendo o melhor deles para realizar a predição para a entrada especificada. BRIAND e WÜST (2001) apresentam uma abordagem para predição de custos de desenvolvimento que utiliza um algoritmo híbrido formado pelos algoritmos *Poisson Regression* e *CART Regression Tree*.

Em alguns trabalhos, técnicas de clusterização foram utilizadas para predição. Em KAUR *et al.* (2010), descreve-se uma abordagem de predição da propensão a falhas usando uma evolução do algoritmo de clusterização *K-means*. Nesse artigo, o *K-means* é utilizado

com outra função de distância (em vez da tradicionalmente usada Distância Euclideana), a distância de Sorensen ou distância de Bray Curtis, sendo referido como algoritmo de clusterização *K-Sorensen-means*. Em SANDHU *et al.* (2010), é apresentada uma metodologia para a predição de propensão a falhas (*fault-proneness*) que utiliza uma técnica de clusterização baseada em densidade (*Density-based Clustering*). São utilizadas métricas de código combinadas com métricas de requisitos. KAUR *et al.* (2010) e KAUR *et al.* (2009) propõem uma metodologia semelhante à do artigo anterior, no entanto utilizam-se, respectivamente, o algoritmo *Fuzzy C Means Clustering* e o algoritmo *K-Means*. Em CHEATHAM *et al.* (1995), é apresentada uma abordagem para a predição do tempo de um teste de software. Essa abordagem utiliza o algoritmo de clusterização COBWEB/3, que é uma evolução do algoritmo COBWEB.

Técnicas híbridas de classificação e regressão são utilizadas em alguns trabalhos. A técnica *Classification and Regression Tree* (CART), que pode ser usada para fins de classificação e de regressão ("S-PLUS 7.0 Guide to Statistics", 2005), aparece em três trabalhos. Em PIATTINI *et al.* (2001), um modelo de predição baseado em um sistema de regras *fuzzy*, denominado *Fuzzy Classification and Regression Tree*, é utilizado. Compõem esse modelo métricas propostas pelo artigo para a avaliação da complexidade interna de Diagramas de Entidade-Relacionamento (DER). GEGICK *et al.* (2008) propõem modelos baseados em CART para a predição da propensão a ataques (*attack-proneness*) em componentes de software. KHOSHGOFTAAR *et al.* (1999) adaptam o CART para definir um modelo que prediz se módulos estão propensos (*fault-proneness*) a terem falhas ou não, após a entrega aos clientes. KUTLUBAY *et al.* (2007) não propõem uma abordagem baseada em CART, porém utiliza um arranjo formado por um algoritmo de classificação e por um de regressão. O artigo apresenta uma abordagem de dois passos para predição de densidade de defeitos. No primeiro passo, módulos são classificados como defectivos e não defectivos. No segundo passo, os dados dos módulos classificados como defectivos são utilizados em uma análise de regressão. Como resultado, a abordagem fornece uma estimativa das densidades de defeitos em módulos defectivos.

As técnicas de regressão são tradicionalmente usadas na predição de valores numéricos. A técnica de regressão *Multivariate Regression* é utilizada em dois trabalhos. Em DAVIDSSON *et al.* (2004), é apresentada uma ferramenta que calcula estimativas sobre a confiabilidade do software através dessa técnica e quantifica o intervalo de confiança da estimativa. KHOSHGOFTAAR *et al.* (1993) também a utilizam, entretanto propõem uma abordagem que melhora seu desempenho através das técnicas Análise de Componentes

Principais e Análise de Cluster, com o objetivo de reduzir a dimensionalidade das variáveis independentes do modelo de predição. Em BHAT *et al.* (2008), é apresentada uma ferramenta, *Tempest*, cuja finalidade é predizer, através de uma técnica de Regressão Estatística, a propensão a falhas de arquivos binários. TARVO (2008) mostra um sistema para predição de erros de regressão (i.e., mudanças indesejadas no comportamento de funcionalidades já estabilizadas do software), o qual utiliza a técnica *Stepwise Logistic Regression* e tem como entradas métricas de código, de dependências e de modificações. Em KAMIYA *et al.* (1999), propõe-se um método para predizer a propensão a falhas (*fault-proneness*) de uma classe em qualquer fase do processo de desenvolvimento do projeto. Em primeiro lugar, definem-se quatro *checkpoints* no processo de desenvolvimento de uma classe, cada um deles correspondendo a uma etapa (ou a um conjunto delas) desse processo. Em seguida, define-se um subconjunto de métricas convencionais aplicáveis ao que é produzido na(s) etapa(s) de cada *checkpoint*. Finalmente, em cada *checkpoint*, estima-se a propensão a falhas da classe usando a técnica *Multivariate Logistic Regression Analysis* com as métricas que se lhe aplicam.

Foram propostas abordagens que utilizam a técnica Redes Neurais para a predição. KHOSHGOFTAAR *et al.* (1992), KHOSHGOFTAAR *et al.* (1994a) e KHOSHGOFTAAR *et al.* (2000) têm em comum o fato de proporem preditores baseados em Redes Neurais. KHOSHGOFTAAR *et al.* (1994b) exploram a aplicação da Análise de Componentes Principais à modelagem de Redes Neurais com vistas a melhorar sua capacidade preditiva. Em KUMAR *et al.* (1998), essa combinação de Redes Neurais e Análise de Componentes Principais também é explorada. THWIN *et al.* (2002) utilizam Redes Neurais (mais especificamente, a técnica Rede de Warp, do tipo *Backpropagation*) para predição do número de falhas. Em QUAH *et al.* (2006), é mostrada uma abordagem para predição através de uma Rede Neural que utiliza uma estratégia de treinamento baseada em algoritmos genéticos.

As abordagens que utilizam a classificação para predição são a maioria do total encontrado. Algumas abordagens aplicam diretamente um algoritmo de classificação para atingir o propósito almejado, como é o caso de DANIEL *et al.* (2008), XING *et al.* (2005), GRAY *et al.* (2009), CATAL *et al.* (2007), PING *et al.* (2000), TOSUN *et al.* (2009a) e MARCHETTO *et al.* (2005), os quais utilizam para a predição, respectivamente, os algoritmos *C4.5*, *Support Vector Machine*, *Support Vector Machine*, *Artificial Immune Recognition System*, *Mixture Model With Expectation-Maximum*, *Naive Bayes* e *k-Nearest Neighbour*. Deve-se incluir também SAHRAOUI *et al.* (2000), que utilizam um algoritmo de árvore de decisão baseado em lógica *fuzzy*, e JALBERT *et al.* (2012), que apresentam uma

abordagem que usa o algoritmo *Support Vector Machine* para predizer a pontuação de mutação[4] (*mutation score*) baseada na combinação de métricas de código e de testes da unidade de compilação em análise. Em outros trabalhos, como os descritos nos parágrafos seguintes, os algoritmos de classificação fazem parte de uma metodologia mais complexa do que a aplicação de um algoritmo já existente.

Em SANDHU *et al.* (2010), para determinar a propensão a defeitos/falhas, a abordagem realiza um pré-processamento para descobrir informações inexistentes sobre as classes dos módulos através do algoritmo de clusterização *K-Means*, utilizando, após a conclusão desse passo, o algoritmo de classificação C4.5 para realizar a predição. Em TOTH *et al.* (2011), são mostrados os resultados da utilização de um *framework* (não detalhado nem referenciado no texto) aplicado ao propósito da predição da complexidade da modificação. A abordagem proposta utiliza uma combinação de métricas de produto e de processo em algoritmos de classificação tradicionais. BAISCH *et al.* (1997) apresentam uma abordagem que utiliza um sistema *Fuzzy* para classificação, gerado com a ajuda de um algoritmo genético. Em SAMI *et al.* (2010), propõe-se um sistema de predição de defeitos que estima o valor de métricas em nível de projeto e dependências com base em métricas de código, todas elas utilizadas como entrada para um Sistema de Classificação *Fuzzy* baseado em Regras (*Rule-based*). Em TOSUN *et al.* (2009b), apresenta-se uma abordagem para a predição de defeitos que utiliza o algoritmo *Naive Bayes*, incorporando, para melhorar seu desempenho, o *framework Call Graph Based Ranking*, o qual gera para cada módulo um valor (*rank*) que é multiplicado por todas as métricas da entrada para ajustar seus valores, de modo que reflitam aspectos das interações intermodulares.

FERZUND *et al.* (2009) propõem uma nova opção de fonte de extração de medidas. No artigo, apresenta-se uma abordagem para a predição de defeitos que utiliza o algoritmo de classificação *Random Forest*, utilizando métricas extraídas de deltas (*hunks*) de revisões do Sistema de Gerência de Configuração. Outra inovação também é oferecida em TIAN *et al.* (2008), no qual se descreve um método de classificação, chamado AODE, para a predição da quantidade de defeitos, a qual é fornecida em categorias (zero defeitos, 1 a 10 defeitos e mais de 10 defeitos).

Das abordagens encontradas pela pesquisa, algumas delas declaram ser *framework*, como as apresentadas em TONU *et al.* (2006) e HADERER *et al.* (2010). Embora pareçam ser semelhantes à abordagem Peixe-Dipnoico, deve-se notar que o conceito de *framework*

[4] Pontuação de mutação é definida como a porcentagem de mutantes que são detectados pela suíte de testes.

utilizado por essas abordagens não se assemelha ao utilizado nesta dissertação. Nas referências encontradas, de acordo com as informações contidas no texto, o termo não se refere a um "conjunto de classes [ou módulos] que compõe um *design* abstrato para uma família de problemas relacionados" (BOSCH, 2000), como considera o Peixe-Dipnoico. Em vez disso, referem-se a uma ferramenta que oferece muitas funcionalidades ao usuário, dentre elas a predição. No caso de TOTH *et al.* (2011), a leitura do texto faz alusão a um conceito de *framework* como o descrito por BOSCH (2000), contudo o artigo não o descreve nem indica qualquer outra referência que viabilize sua análise.

3.7 ANÁLISE QUANTITATIVA

A partir dos registros das informações obtidas das publicações selecionadas na 3ª etapa, também foram realizadas Análises Quantitativas das publicações, baseadas nas questões secundárias do estudo. Essas análises auxiliaram e subsidiaram considerações com o intuito de discutir os resultados da busca com relação à questão de pesquisa.

É importante ressaltar que, devido à variação nas nomenclaturas, as respostas a algumas questões secundárias obtidas das publicações foram agrupadas de acordo com suas equivalências. Por exemplo, na questão secundária referente à finalidade da técnica (QS4), respostas como Predição de Propensão a Falhas, Predição de Defeitos, Predição do Número de Erros, Predição do Número de Falhas e similares passaram a ser referidas como Predição de Defeitos/Falhas. Essa decisão foi tomada para evitar que se gerassem estatísticas de baixo valor significativo para respostas diferentes que se referissem à mesma entidade.

Considerando-se a primeira questão secundária do estudo (*QS1: De onde são obtidos os dados para a predição?*), os resultados foram agrupados de acordo com as fontes de extração das medidas pelas técnicas. Cada agrupamento tem uma semântica específica, conforme descrito a seguir:

- Projeto de Software: a abordagem extraiu medidas a partir da análise de um conjunto de unidades de compilação não submetido a gerenciamento de configuração. Por exemplo, projetos compilados com *Maven*[5], *Ant*[6] ou *Make*[7].
- Bases de Medidas: a abordagem extraiu medidas a partir da análise de uma base de dados contendo registros de módulos acompanhados dos valores de

[5] http://maven.apache.org/
[6] http://ant.apache.org/
[7] http://www.gnu.org/software/make/

métricas calculados. Por exemplo, as bases de medidas do *NASA Metric Data Program* [8](MDP).

- Sistema de Gerência de Configuração: a abordagem extraiu medidas a partir da análise de um conjunto de unidades de compilação submetidas a gerenciamento de configuração em sistemas como Subversion[9] e Git[10].
- Diagrama: a abordagem extraiu medidas a partir da análise de diagramas usados para representação conceitual ou estrutural do software. Por exemplo, diagramas UML e Entidade-Relacionamento.
- Arquivos Binários: a abordagem extraiu medidas a partir da análise de arquivos compilados e prontos para a execução em computadores.
- Casos de Teste: a abordagem extraiu medidas a partir da análise de unidades de compilação onde testes unitários estão codificados. Por exemplo, os casos de testes desenvolvidos com o framework JUnit[11].

Como pode ser observado na Figura 11, na maioria absoluta dos casos, em aproximadamente 75% do total, os dados foram obtidos de Projetos de Software ou de Bases de Medidas. Os trabalhos que mais se assemelham à abordagem proposta por essa dissertação em relação à QS1, isto é, cujos dados foram obtidos de um Sistema de Gerência de Configuração, correspondem a apenas 7% do total.

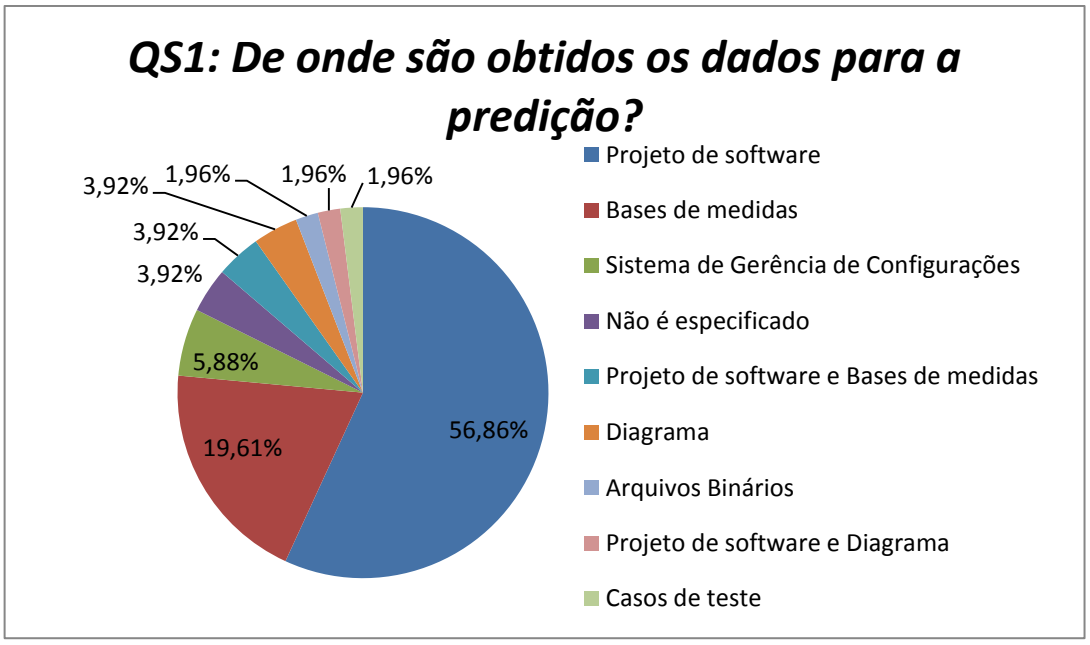

Figura 11. Respostas à primeira questão secundária de pesquisa (QS1).

[8] http://nasa-softwaredefectdatasets.wikispaces.com/
[9] http://subversion.apache.org/
[10] http://git-scm.com/
[11] http://junit.org/

As respostas à segunda questão secundária do estudo (*QS2: Qual a granularidade dos artefatos de software analisados para construir a predição?*) estavam relacionadas à granularidade dos artefatos de software dos quais as abordagens pesquisadas extraíram medidas para uso em seus algoritmos de predição. Em outras palavras, buscou-se identificar através dessa questão secundária qual o menor tipo de elemento de programação a que as medidas utilizadas para a predição se referem. Nos casos em que a resposta listava mais de uma granularidade, foi considerada a menor. Por exemplo, se a resposta era "Classe, Método", escolheu-se "Método".

Na maioria das respostas à QS2, o termo encontrado aparece nos artigos como um conceito de senso comum, não sendo definido. É o caso dos termos módulo, classe, método/função e componente. É importante ressaltar que apesar de, segundo o Glossário Padrão de Terminologias de Engenharia de Software (IEEE, 1990), o termo componente ser sinônimo de módulo, ele foi mantido na análise quantitativa, já que no artigo onde aparece não é especificada a relação entre eles. A resposta "Delta (*hunks*) de revisões do SGC" se refere a uma unidade de modificação, gerada pela adição ou remoção de um conjunto de linhas de código (FERZUND; AHSAN; WOTAWA, 2009). A resposta "Entidade de diagrama" se refere a elementos próprios de diagramas de projeto de software, como entidades, classes e associações.

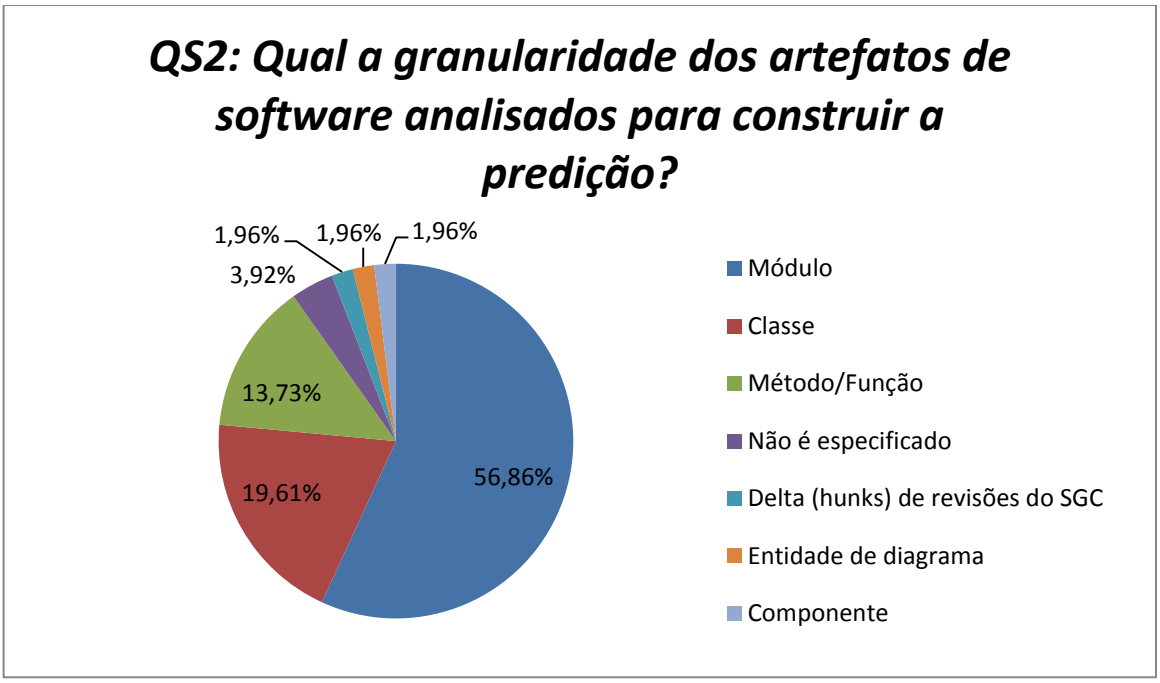

Figura 12. Respostas à segunda questão secundária de pesquisa (QS2).

Observando a Figura 12, percebe-se que a maioria absoluta das abordagens obtém medidas de características estruturais em um nível elevado de granularidade (Módulo ou Classe), totalizando aproximadamente 75% dos casos. A abordagem proposta por esta dissertação também está focada nesse nível.

As respostas à terceira questão secundária de pesquisa (*QS3: Que tipos de medidas estruturais são usados?*) foram agrupadas nas seguintes categorias de métricas:

- Métricas de Código: métricas que analisam atributos e construções sintáticas presentes em qualquer código-fonte. Tais construções são, por exemplo, o Número de Operadores/Operandos, Número de Instruções, Número de Linhas de Código, Complexidade Ciclomática (que analisa instruções de desvio do fluxo de execução), Número de Parâmetros, etc.
- Métricas OO: métricas que analisam atributos e construções sintáticas presentes somente em códigos-fontes escritos sob o paradigma de orientação a objetos. São exemplos dessa categoria: *Depth of Inheritance*, *LackOfCohesionInMethods*, Número de Métodos Sobrescritos, Número de Descendentes, Número de Atributos Públicos/Privados/Protegidos, *Weighted Method Count*, entre outras.
- Métricas de Diagrama: métricas relacionadas a entidades próprias de diagramas conceituais ou de implementação, como Número de Entidades, Número de Associações, etc.
- Métricas Específicas: métricas nunca propostas na literatura, definidas no contexto de um dos artigos lidos.
- Métricas de Dependência: métricas que medem o nível de dependência ou acoplamento entre elementos de programação, como *Fan-In*, *Fan-Out*, Acoplamento entre Objetos, *Lack of Cohesion in Methods*, Número de Dependências Externas, Número de Dependências Internas, entre outras.

A Figura 13 traz um histograma com a contagem das categorias encontradas nas respostas. As métricas de código são as mais utilizadas nos artigos analisados. Vale ressaltar que grande parte dos artigos utiliza mais de uma categoria de métrica, portanto em alguns deles, por exemplo, somente as métricas de código são utilizadas, enquanto em outros elas são utilizadas em conjunto com alguma(s) de outra(s) categoria(s). O Peixe-Dipnoico utiliza as métricas fornecidas pela biblioteca Oceano (ver Capítulo 4), desenvolvida pelo Grupo de Evolução e Manutenção de Software (GEMS) da Universidade Federal Fluminense, a qual em

sua versão atual possui a implementação das categorias de métricas “Métricas de Código” (Número de Linhas de Código, Complexidade Ciclomática, etc.), “Métricas OO” (Número de Métodos Públicos, Número de Atributos Privados, etc.) e “Métricas de Dependência” (*Lack of Cohesion Among Methods*, métricas do software de medições *Dependometer*[12], etc.).

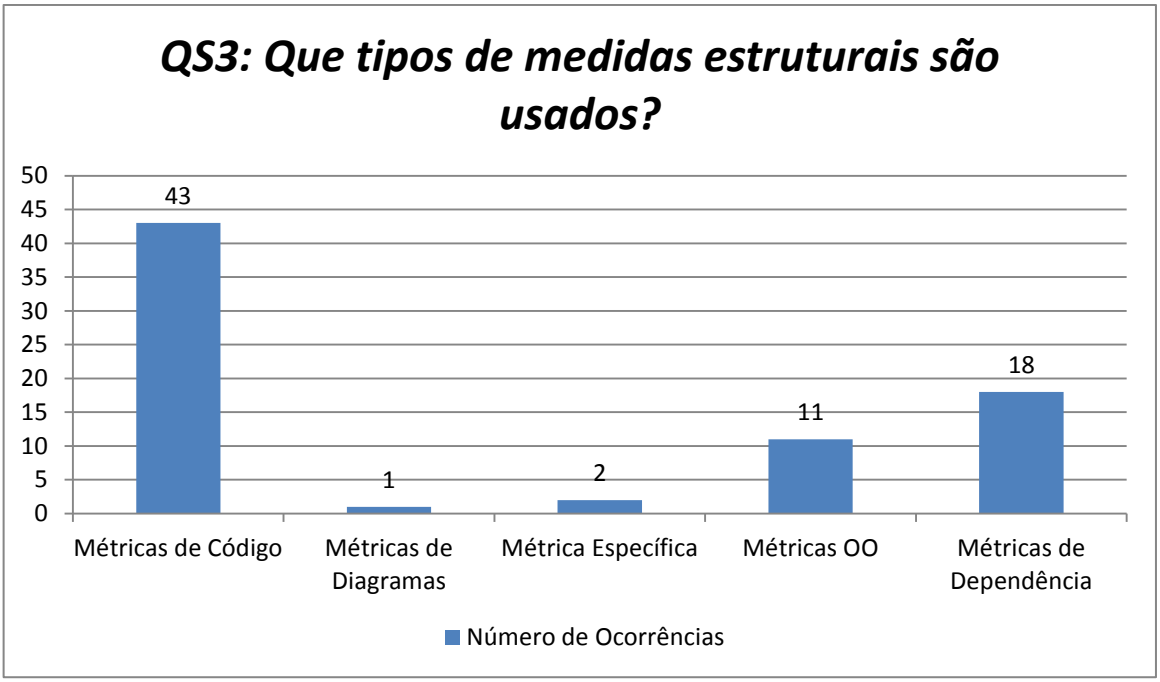

Figura 13. Respostas à terceira questão secundária de pesquisa (QS3).

As respostas à quarta questão secundária de pesquisa do estudo (*QS4: Para que propósito a abordagem é destinada?*) revelam o que as abordagens encontradas se propõem a predizer. Por exemplo, algumas se destinam a predizer informações sobre defeitos e falhas (predição do número de falhas, propensão a falhas/defeitos), enquanto outras se destinam a predizer informações sobre a qualidade do software (predição da confiabilidade ou da manutenibilidade do software na próxima *release*). Nas legendas do gráfico da Figura 14, antes de todos os rótulos deve-se incluir a expressão “predição de”, a qual, por restrições de espaço no gráfico e para melhorar sua visualização, foi omitida.

O gráfico da Figura 14 mostra que os principais objetivos dos trabalhos relacionados à predição usando características estruturais se dividem em Predição de Defeitos/Falhas (propensão a falhas, número de defeitos, etc.) e em Predição de Qualidade (nível de manutenibilidade, testabilidade, confiabilidade, etc.), totalizando 65% dos casos. A abordagem apresentada por esta dissertação se enquadra no objetivo de Predição de Métricas

[12] http://source.valtech.com/display/dpm/Dependometer

de Código, o qual, no gráfico, figura em apenas 2% dos trabalhos obtidos na pesquisa bibliográfica.

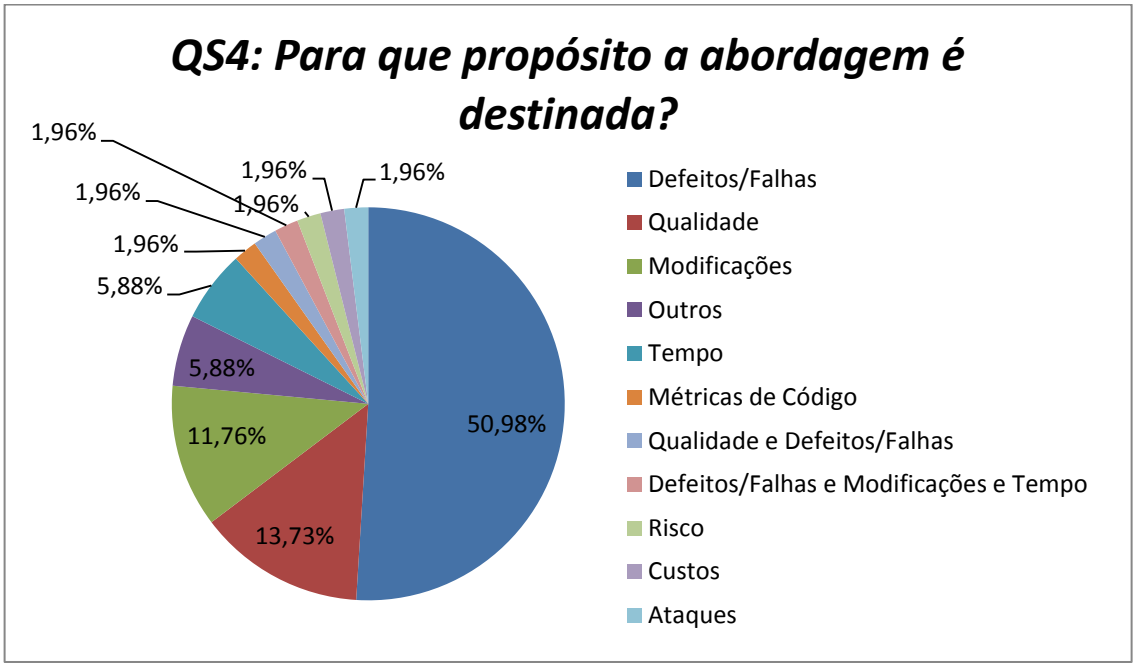

Figura 14. Respostas à quarta questão secundária de pesquisa (QS4)

As respostas à quinta questão secundária (*QS5: Qual a técnica utilizada para construir a predição?*) do estudo identificam o nome da técnica de predição utilizada. Como os nomes são muito variados, foram agrupados em categorias com a semântica descrita a seguir:

- Classificação: abordagens que utilizam algoritmos de aprendizagem de máquina que, a partir de atributos de uma instância, determinam (ou predizem) a que classe tal instância pertence (MAIMON; ROKACH, 2010).
- Clusterização: abordagens que utilizam algoritmos de aprendizagem de máquina que agrupa instâncias com atributos similares (MAIMON; ROKACH, 2010).
- Regressão: abordagens que utilizam algoritmos que inferem o valor de um atributo a partir da análise dos valores de outros atributos (GUJARATI, 2003).
- Redes Neurais: algoritmos de aprendizagem de máquina que simulam o funcionamento de neurônios biológicos (MAIMON; ROKACH, 2010). Podem ser usados para fins de classificação ou regressão. Por aparecer em uma quantidade significativa das abordagens pesquisadas, decidiu-se considerá-lo uma categoria à parte.

- Análise de Séries Temporais: conjunto de modelos de predição baseados em séries temporais (sequências de valores registrados ao longo do tempo) (GUJARATI, 2003).
- Compostas: no contexto da pesquisa bibliográfica realizada, são abordagens compostas por mais de uma técnica, escolhendo para fornecer a predição aquela que apresentar individualmente o melhor resultado (abordagem dinâmica) ou combinando as funcionalidades de todas as técnicas disponíveis (abordagem híbrida). Uma abordagem desse tipo não é contabilizada na categoria das técnicas que utiliza como base, isto é, se ela utiliza técnicas de classificação e de regressão, somente a contagem da categoria "Composta" é incrementada, permanecendo as contagens das categorias "Classificação" e "Regressão" inalteradas.
- Técnica Específica: definidos no contexto da pesquisa bibliográfica realizada, são metodologias, algoritmos ou modelos de predição inovadores, não propostos anteriormente na literatura.
- Outros: no contexto da pesquisa bibliográfica realizada, são abordagens já propostas na literatura que não se encaixam em uma das categorias anteriormente listadas.

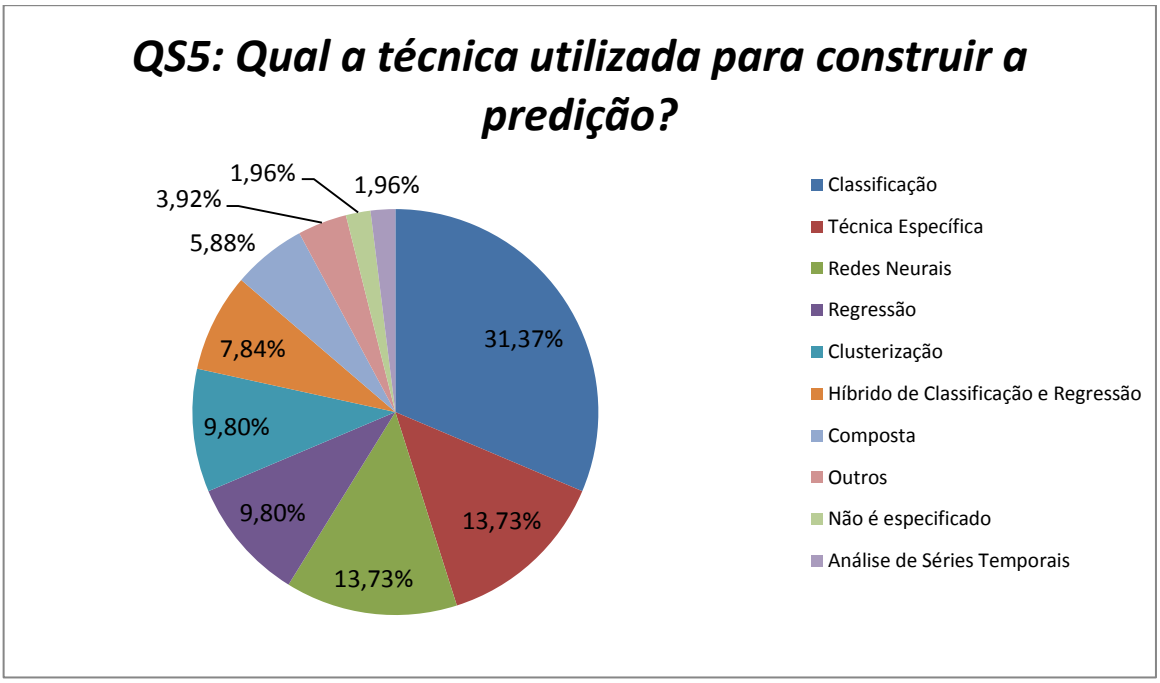

Figura 15. Respostas à quinta questão secundária de pesquisa (QS5).

A observação do gráfico da Figura 15 revela que não existe uma hegemonia de uma categoria de técnicas sobre as outras. Em 31% dos casos, a categoria de técnicas utilizada foi a Classificação. Em seguida, está a categoria Técnica Específica com 14%, empatada com a categoria Redes Neurais. A abordagem Peixe-Dipnoico se enquadra em mais de uma categoria. Ela tem potencial para se enquadrar em qualquer categoria que se propuser, haja vista o fato de ser um *framework* para implementação e experimentação de algoritmos de predição, como explicado no Capítulo 4. Em sua versão atual, dados os algoritmos já implementados tendo-o como base, ela pode ser incluída nas categorias Técnica Específica, Regressão, Dinâmica e Análise de Séries Temporais.

Para as respostas à sexta questão secundária de pesquisa (*QS6: Quais métricas de desempenho foram utilizadas para avaliar a abordagem?*), duas análises foram realizadas. A Figura 17 mostra a distribuição da frequência das métricas de desempenho de acordo com o seu uso na QS7. É possível observar que em 16% das publicações analisadas não foi possível saber o desempenho da abordagem proposta, sendo esse valor composto pela soma das porcentagens das categorias “Não houve validação experimental” e “Não é especificado”. Na Figura 16, apresentam-se as frequências de uso de todas as métricas de desempenho encontradas nas publicações. Em algumas publicações, foi utilizada mais de uma métrica. Dessa forma, observa-se que a métrica mais frequente é a Acurácia, seguida pela Cobertura e pela Taxa de Falsos Positivos (comumente chamada de Taxa de Alarmes Falsos). A categoria “Outros”, que aparece como uma das mais frequentes, refere-se, na verdade, a um conjunto de métricas que individualmente possuem frequência de uso nos trabalhos muito baixa ou que possuem uma semântica específica. São exemplos dessa categoria o Coeficiente de Múltipla Determinação (R^2) e o Coeficiente de Correlação (r) (GUJARATI, 2003).

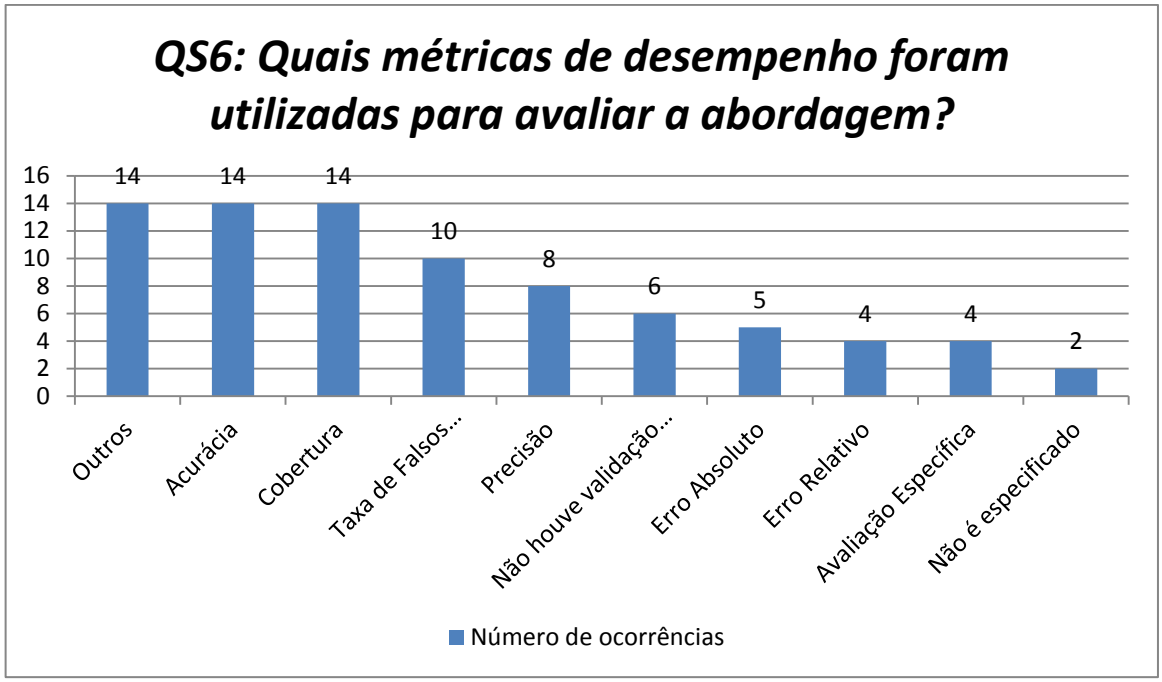

Figura 16. Respostas à sexta questão secundária de pesquisa (QS6).

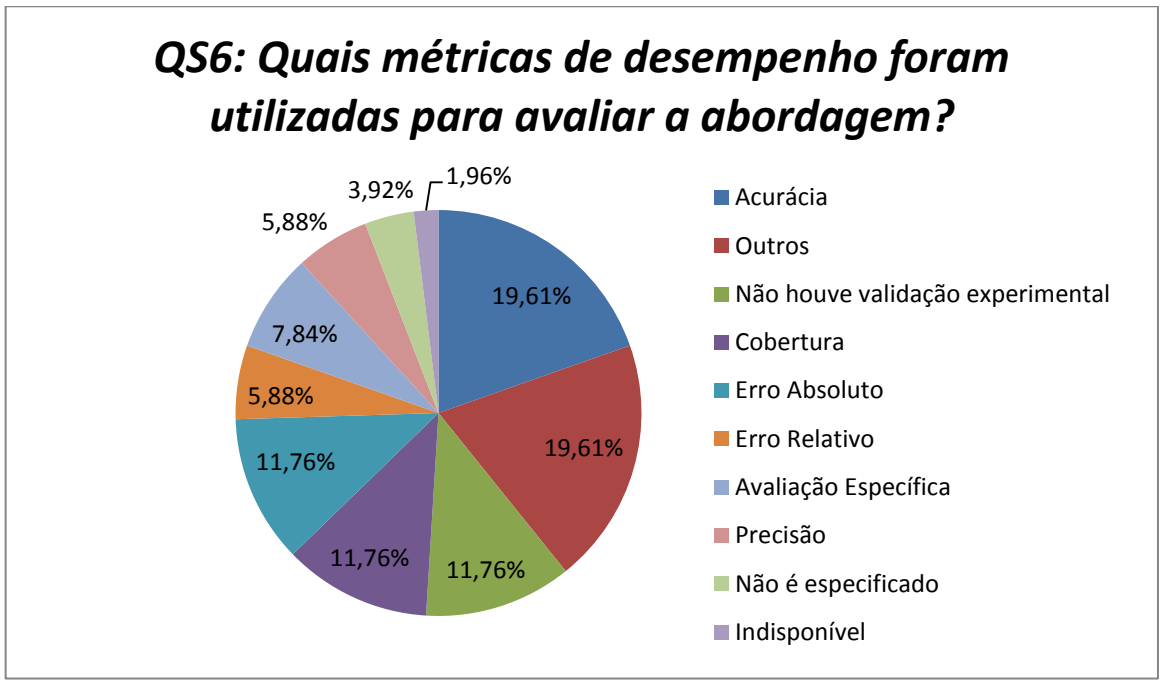

Figura 17. Distribuição da frequência das respostas a QS6 de acordo com o seu uso na QS7.

Nas respostas à sétima questão secundária de pesquisa (*QS7: Qual é o desempenho aproximado das predições fornecidas pela abordagem?*), como a maioria das abordagens utiliza mais de uma métrica de desempenho e não há métricas comuns a todos os trabalhos, as métricas de desempenho a que se referem são diversas, não possibilitando uma comparação

uniforme. Por esse motivo, será apresentada a faixa de variação para cada métrica de desempenho utilizada, acompanhada da média e desvio-padrão dos valores, exceto para a métrica Erro Absoluto, para a qual, devido ao alto desbalanceamento entre as magnitudes, somente serão listados os valores encontrados.

- Cobertura
  - Faixa de valores: 50% a 100%
  - Quantidade de amostras: 6
  - Média: 89,66%
  - Desvio-padrão: 19,9%
- Acurácia
  - Faixa de valores: 56% a 93%
  - Quantidade de amostras: 10
  - Média: 80,49%
  - Desvio-padrão: 11,59%
- Precisão
  - Faixa de valores: 43% a 83%
  - Quantidade de amostras: 3
  - Média: 66,1%
  - Desvio-padrão: 20,5%
- Média dos Erros Relativos
  - Faixa de valores: 9,9% a 39,8%
  - Quantidade de amostras: 3
  - Média: 24,27%
  - Desvio-padrão: 13,9%
- Média dos Erros Absolutos: 0,1656; 0,7873; 4,95; 5,1; 1965 e 10.000

As respostas à oitava questão secundária desse estudo (*QS8: Oferece apoio ferramental?*) revelam se as abordagens apresentadas ofereciam apoio ferramental, isto é, se ofereciam algum meio que facilitasse o uso da abordagem pelos usuários. O oferecimento de aplicativos e/ou *frameworks* que implementem a abordagem descrita é um desses meios. O Peixe-Dipnoico oferece apoio ferramental, assim como o faz um pequeno grupo correspondente a 13% das abordagens pesquisadas.

3.8 CONSIDERAÇÕES FINAIS

Ao longo deste capítulo, foi delineado um processo de execução de um estudo baseado em Revisão Sistemática da Literatura. A partir de um protocolo de pesquisa que buscou o máximo equilíbrio possível entre precisão (retorno de referências relevantes) e cobertura (retorno das referências com potencial relevância), o processo que se seguiu reduziu as chances de que a pesquisa bibliográfica fosse tendenciosa ou insuficiente. Além disso, a descrição dos passos tomados para a realização desse estudo possibilita que outros pesquisadores repliquem os resultados obtidos.

Os resultados obtidos mostram que existem muitas abordagens e muitos estudos com foco em obter o melhor desempenho possível nas predições. A predição relacionada a aspectos do software é um tema de crescente importância e notoriedade, haja vista os benefícios que pode trazer aos processos de desenvolvimento, como a redução de custos e a mitigação de riscos. Entretanto, apesar da quantidade de pesquisas nesse tema, não existe, dentro do campo de visão da pesquisa bibliográfica realizada nesta dissertação, uma proposta que ajude a melhorar a produtividade das pesquisas no tema predição. Pouco ainda se fez com o objetivo de difundir entre os profissionais, através de ferramentas ou *frameworks*, as capacidades alcançadas com as pesquisas já publicadas. Dessa forma, uma proposta que atenda a essas demandas levantadas anteriormente tem potencial para trazer uma significativa contribuição nas pesquisas e no uso de predições no contexto do desenvolvimento de software.

O Capítulo 4 desta dissertação traz a proposta de um *framework* que propicia a experimentação tanto de técnicas de predição novas quanto de técnicas tradicionais. Através de pontos de extensão e de uma arquitetura reutilizável, em sua versão atual o *framework* auxilia o processo de modelagem de novos algoritmos de predição, viabiliza a utilização de outros tipos de repositórios de software e permite a elaboração de outras formas de experimento (além das que já estão implementadas). Além disso, possibilita a criação de diversos tipos de aplicações na área de predição de software, como é mostrado no Capítulo 4.

CAPÍTULO 4 – PEIXE-DIPNOICO

4.1 INTRODUÇÃO

No Capítulo 3, foram apresentados 51 trabalhos relacionados ao tema predição em software com foco em suas características estruturais. Todos visavam melhorar o desempenho de predições através de novas abordagens ou do aprimoramento de abordagens preexistentes. O capítulo também mostrou que a predição relacionada a aspectos do software é um tema de crescente importância e notoriedade, haja vista os benefícios que pode trazer aos processos de desenvolvimento, como a redução de custos e a mitigação de riscos. Entretanto, devido à ausência de um padrão nas metodologias e nas ferramentas adotadas, muito retrabalho é realizado entre as pesquisas, como nos processos de obtenção de medidas, na construção de experimentos e na apresentação de resultados. Todas essas atividades são comuns a esse tipo de pesquisa, portanto esforços que visem tratar o domínio de forma mais abstrata, oferecendo mecanismos de reutilização dos processos mencionados, poderiam oferecer produtividade a essas pesquisas. Não há, entretanto, de acordo com a revisão da literatura realizada, uma proposta que atenda a esses objetivos. Dessa forma, o presente capítulo traz a proposta de um *framework*, o Peixe-Dipnoico, que propicia a experimentação tanto de técnicas de predição novas quanto de abordagens tradicionais. Através de pontos de extensão e de uma arquitetura reutilizável, o *framework* auxilia na modelagem de técnicas de predição, na utilização de vários tipos de repositórios de software, na elaboração de experimentos para avaliar o desempenho das técnicas modeladas e na geração de saídas para os resultados dos experimentos.

De acordo com BOSCH (2000), um *framework* é um conjunto de classes que compõe uma arquitetura abstrata para a modelagem de soluções que atendam a uma família de problemas relacionados. No Peixe-Dipnoico, sua arquitetura atende a problemas relacionados ao tema predição em software. Seu projeto é abstrato, já que oferece pontos de extensão e propicia a reutilização de processos comuns ao domínio, o que o torna habilitado a ser útil a várias aplicações, sendo uma delas a que é mostrada na Seção 4.7.

A arquitetura do Peixe-Dipnoico possui 5 elementos principais: (1) Metamodelo, (2) Controlador de Entrada, (3) Controlador de Saída, (4) *Kernel* e (5) Técnica de Predição. Além desses, outros elementos relacionados funcionam como pontos de extensão, como é o caso de *Métrica*, *Sistema de Controle de Versões* e *Formatos de Saída*. *Técnica de predição* também é

um ponto de extensão, como mostrado nos parágrafos seguintes. A Figura 18 mostra um diagrama de componentes que representa o núcleo da arquitetura do Peixe-dipnoico.

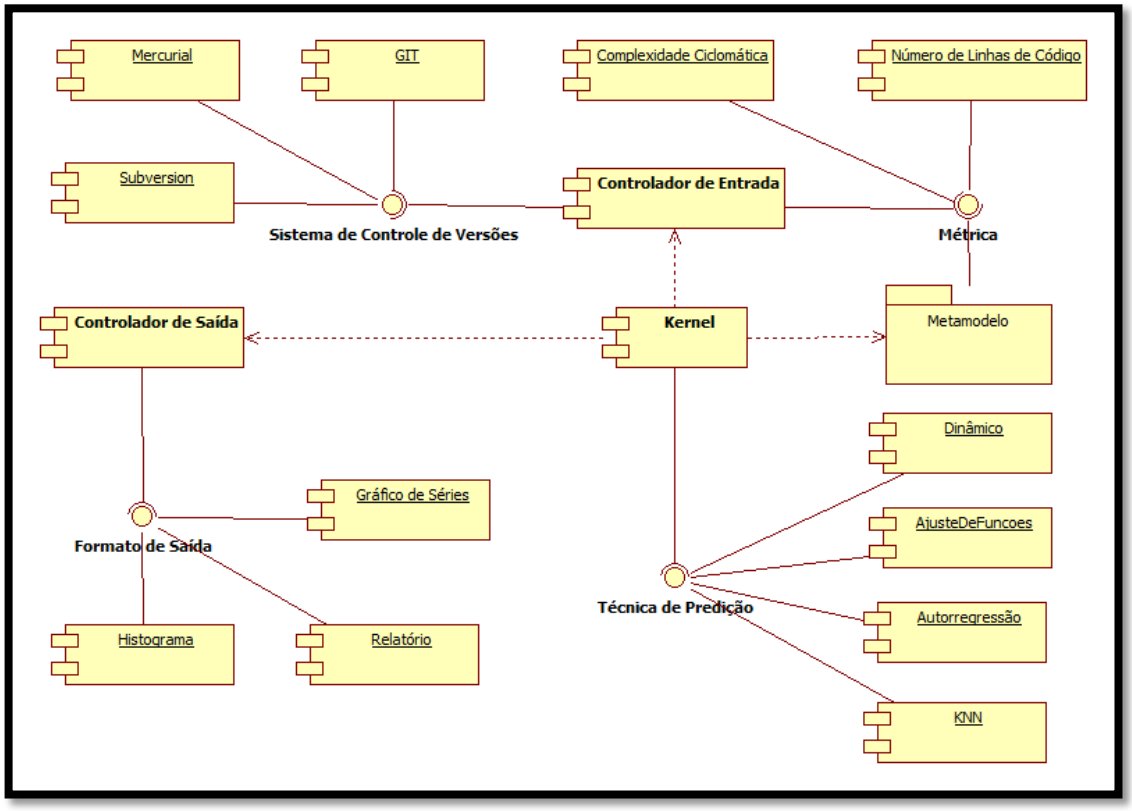

Figura 18. Arquitetura do framework Peixe-Dipnoico.

Na arquitetura do Peixe-Dipnoico existe um *Kernel* no centro, o qual orquestra sua execução. O *Kernel* recebe do *Controlador de Entrada* séries temporais lidas de um repositório de um sistema de controle de versões. Cada série se refere a um artefato (classes de um projeto de software) e é formada por valores resultantes de um processo de medição de cada revisão desse artefato. Recebidas as séries, o *Kernel* as utiliza em experimentos ou diretamente como entradas para técnicas de predição. No contexto desta dissertação, um experimento é um processo de medição do desempenho de uma técnica, no qual os resultados das predições geradas para um conjunto de séries utilizadas como entrada para essa técnica são comparados a valores medidos para cada uma dessas séries. Os resultados desses experimentos ou das predições geradas por essas técnicas de predição são encaminhados pelo *Kernel* ao *Controlador de Saída* para exibição em um dos formatos que este disponibiliza.

Os pontos de extensão do Peixe-Dipnoico são quatro: *Técnica de Predição*, *Sistema de Controle de Versões*, *Métrica* e *Formato de Saída*. Cada um deles é explicado a seguir.

- Técnica de Predição: representa as técnicas de predição que venham a ser modeladas através do *framework*. Esse ponto de extensão possibilita, através

do mecanismo de herança, que novas técnicas sejam incorporadas aos experimentos e às aplicações previamente definidos sem a necessidade de realizar modificações na estrutura do *framework*.

- Sistema de Controle de Versões: auxilia na especialização do processo de obtenção de séries, de modo que seja possível, expressando mais concretamente, extrair medidas de artefatos armazenados em diferentes sistemas de gerência de configuração, como *GIT* e *Subversion*.
- Métrica: representa as métricas a serem usadas pelo *framework*. Através do mecanismo de herança, novos extratores de métricas podem ser implementados para gerar dados sob novas perspectivas.
- Formato de Saída: auxilia na definição de novos formatos de visualização dos dados dos experimentos e das execuções das técnicas.

O componente *Controlador de Saída* oferece formas diversificadas (ponto de extensão *Formato de Saída*) de visualizar os resultados das predições. Esse componente realiza a geração de saída dos resultados dos experimentos, por meio de diferentes tipos de gráficos, como gráficos de séries e histogramas, ou por meio de relatório.

Neste capítulo, a Seção 4.2 apresenta as entidades que compõem o metamodelo do *framework*. Em seguida, são apresentados as funcionalidades e os objetivos dos componentes *Controlador de Entrada* (Seção 4.3), *Controlador de Saída* (Seção 4.4) e *Kernel* (Seção 4.5). A Seção 4.6 apresenta e descreve o funcionamento das técnicas de predição desenvolvidas nessa versão do *framework*, a qual disponibiliza os quatro tipos de técnicas mostrados na Figura 18, que são: KNN, Autorregressão, Ajuste de Funções e Dinâmico. A Seção 4.7 apresenta exemplos de utilização prática do Peixe-Dipnoico. Por fim, a Seção 4.8 encerra o capítulo fazendo uma análise da proposta apresentada.

4.2 METAMODELO

No *framework* proposto, o pacote *Metamodelo* é composto por entidades passivas, que primordialmente armazenam dados, servindo de base para todos os demais componentes da arquitetura. Essas entidades representam os conceitos de Série Temporal, Experimento e Resultado de experimento.

O conceito básico de Série Temporal é representado pela entidade *Serie*. No *framework*, ela é uma classe contendo uma lista de valores (*sequência*). Além dessa lista, possui operações de manipulação dos dados para atender a algumas demandas da pesquisa,

mostradas ao longo da seção. A Figura 19 mostra em diagrama o conceito de Série Temporal no contexto da dissertação.

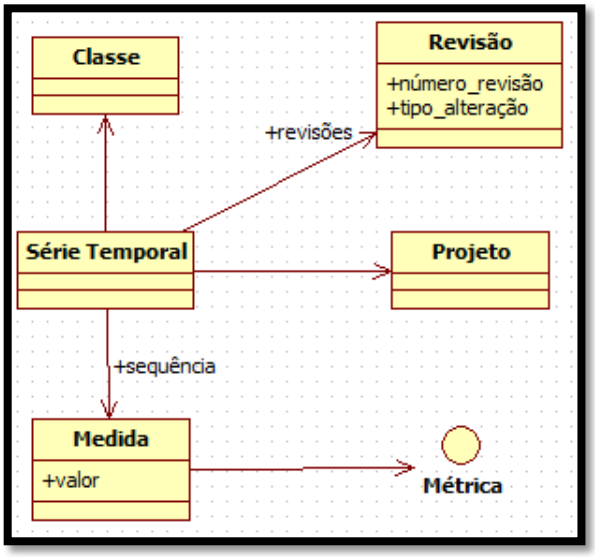

Figura 19 – Série Temporal.

O modelo de representação das séries possui identificações dos números das revisões (*numero_revisão*) e do tipo de modificação (*tipo_alteração*) realizado em cada *commit* (i.e., adição, remoção, modificação, etc.), além de informações sobre o nome do artefato, do projeto de onde foi obtido e da métrica a que se referem seus valores. Esses registros são importantes para facilitar a obtenção dessas informações durante a geração de saída (Seção 4.4) e tornar as classes que lidam com essas informações mais coesas.

A menor granularidade das medidas armazenadas nas séries é a de **classe**. Essa decisão foi guiada por restrições das ferramentas utilizadas para a medição dos códigos-fontes e dos Sistemas de Controle de Versão, que, em geral, gerenciam versões de arquivos. Embora já exista uma abordagem que gerencia versões de métodos, construtores e campos (HATA; MIZUNO; KIKUNO, 2011), esta ainda não é difundida, não sendo viável, portanto, realizar medições em artefatos gerenciados por Sistema de Controle de Versões que estejam nesse nível de granularidade.

O processo de obtenção de séries, descrito na Seção 4.3, é precedido pela atividade de medição de atributos do software. Nessa atividade, revisões das classes pertencentes a um projeto de software são submetidas uma a uma a um algoritmo que calcula a dimensão (medida) de um desses atributos. A definição dos atributos e a forma como suas dimensões são calculadas são estabelecidas pelas métricas, mostradas na Seção 2.2. Variadas métricas podem ser implementadas e adicionadas ao *framework*, através do ponto de extensão *Métrica*, como pode ser visto na Seção 4.3.

Cada medida armazenada em uma série é a representação do valor resultante da medição de uma determinada revisão de uma classe. Dessa forma, a dimensão tempo, comum em séries temporais, não foi modelada pelo *framework*. Essa decisão foi tomada para evitar a ocorrência de medidas repetidas nas séries, o que facilmente acontece em classes com baixa frequência de modificações. O registro desses valores repetidos gera um volume massivo de informações que poderiam se resumir apenas ao registro dos novos valores gerados. Além disso, não é a passagem do tempo que modifica a medida do artefato armazenado no repositório, mas sim a efetivação de um *commit*. Para ilustrar a diferença entre as duas formas de representação do tempo explicadas neste parágrafo, a Figura 20 e a Figura 21 mostram a representação de uma mesma série, a primeira baseada em *commit* e a segunda baseada em tempo cronológico.

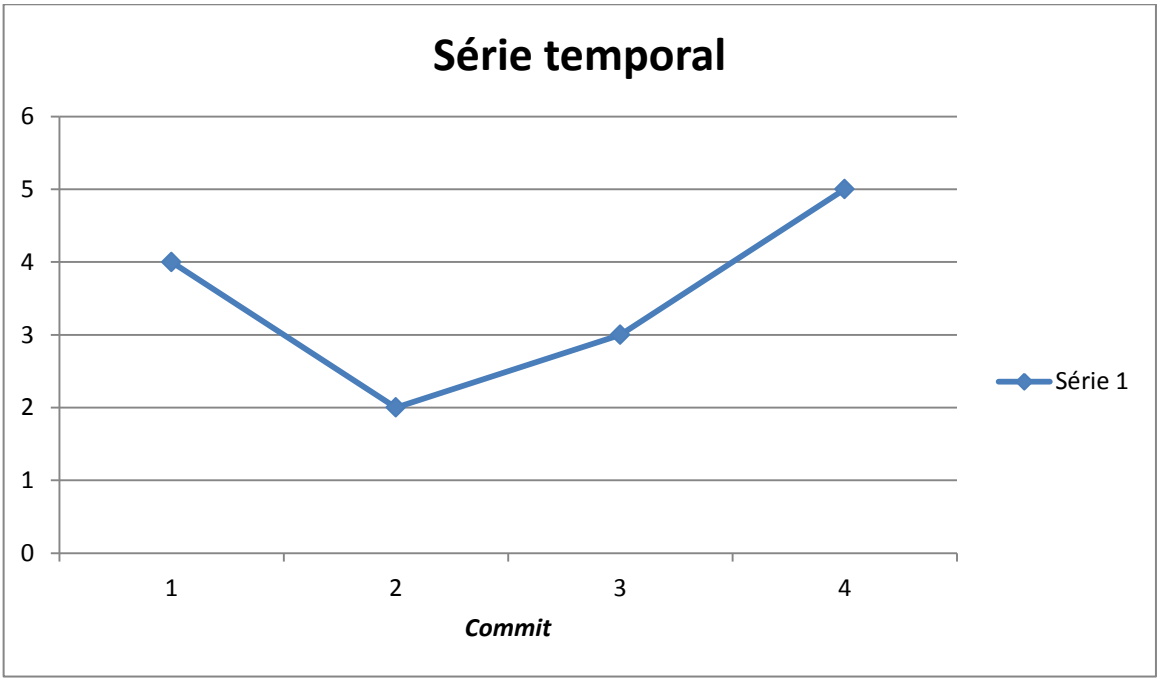

Figura 20. Série temporal baseada em *commit*.

Sendo o Peixe-Dipnoico um *framework* com foco em experimentos, a modelagem de experimentos tornou-se necessária. A entidade *Experimento* representa a essência de um experimento sob a ótica da predição de séries temporais. Dessa forma, conforme mostrado pela Figura 22, uma instância dessa entidade se relaciona com todas as entidades do metamodelo. Essa instância também se relaciona com o componente *Técnica de Predição*, registrando, no início de seu ciclo de vida, uma lista de séries temporais (*series*), usadas uma a uma iterativamente em um processo, descrito a seguir, por uma técnica de predição (*técnica*).

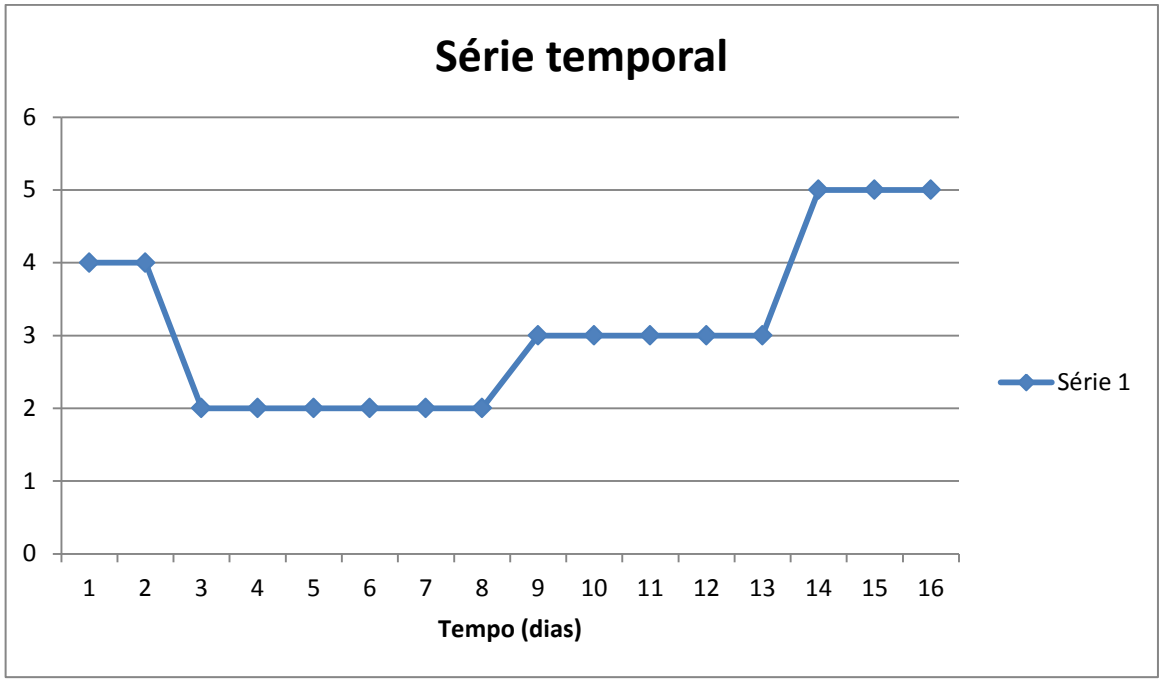

Figura 21. Série temporal baseada em tempo cronológico.

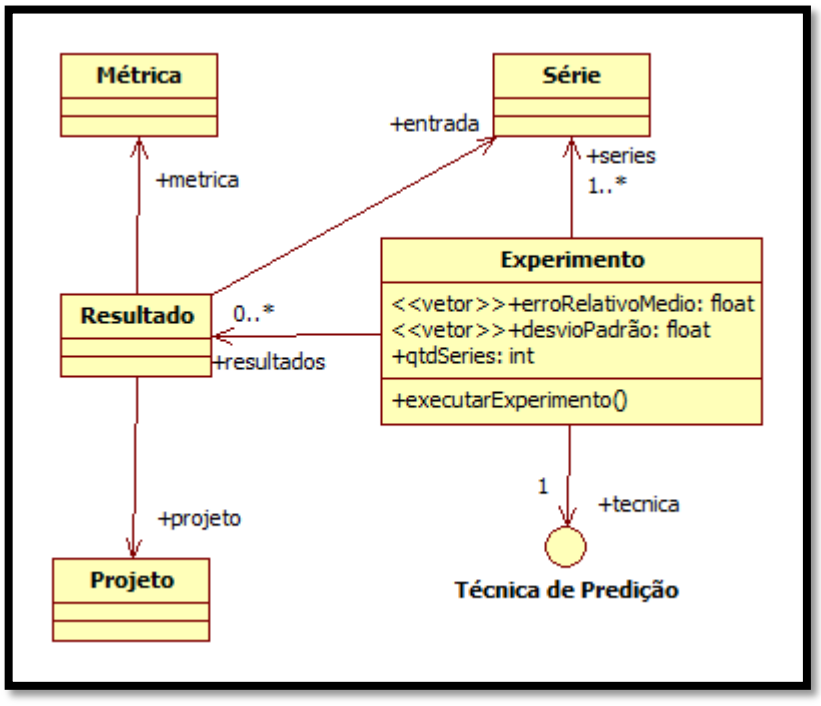

Figura 22. Experimento.

No contexto das técnicas de predição desta abordagem, um experimento, tendo à disposição as séries, possui o seguinte fluxo de execução (Figura 23), presente no método *executarExperimento*:

1. Obter a próxima série temporal.

2. Dividir a série temporal em duas partes, sendo a primeira parte a entrada para a técnica de predição e a segunda parte uma série composta de valores utilizados para comparação com os valores preditos, chamada de *série efetiva*. Os valores da série efetiva são os últimos da série principal e seu tamanho é definido pelo horizonte de predição, que consiste no número de instantes futuros para os quais se realizam predições. Por exemplo, para a série S = {1,2,3,4,5}, se o horizonte é 2, a série efetiva será {4,5}.
3. Executar a técnica de predição com a entrada definida no passo 2.
4. Armazenar na entidade *Resultado* a série efetiva e a série composta pelos valores preditos.
5. Se existirem mais séries a serem obtidas, voltar ao passo 1. Caso contrário, encerrar o processo.

Como mostrado na Figura 23, os experimentos possuem resultados. Esses resultados precisam ser guardados para que um pesquisador que utilize o *framework* possa analisar o comportamento da sua técnica implementada e obter medidas de desempenho globais, com base nos resultados individuais do experimento. Para suprir essa demanda, a entidade *Resultado* armazena esses resultados (erro absoluto e erro relativo), as informações sobre o experimento, o nome e os parâmetros da técnica de predição e informações sobre a série de entrada. Essas informações são utilizadas pelo *Controlador de Saída* na geração de relatórios e gráficos. Por exemplo, a série predita e a série efetiva são exibidas nos gráficos de série em sobreposição à série de entrada utilizada na técnica, bem como os erros de cada predição, exibidos nos relatórios também gerados pelo *Controlador de Saída*.

4.3 CONTROLADOR DE ENTRADA

O *Controlador de Entrada* é o componente responsável pela obtenção de séries temporais para serem utilizadas pelas técnicas de predição e pelos experimentos. Esse componente é constituído por dois dos pontos de extensão do *framework*: a interface *Sistema de Controle de Versão* da Figura 18 e a interface *Métrica*. A primeira possibilita a implementação de formas diferentes de obter as séries temporais e de selecionar as medidas que venham a fazer parte delas. A segunda possibilita a implementação de vários tipos de extratores de medidas, que definem o processo de medição de várias métricas conhecidas, como Número de Linhas de Código e Complexidade Ciclomática. A Figura 24 mostra a estrutura desse componente.

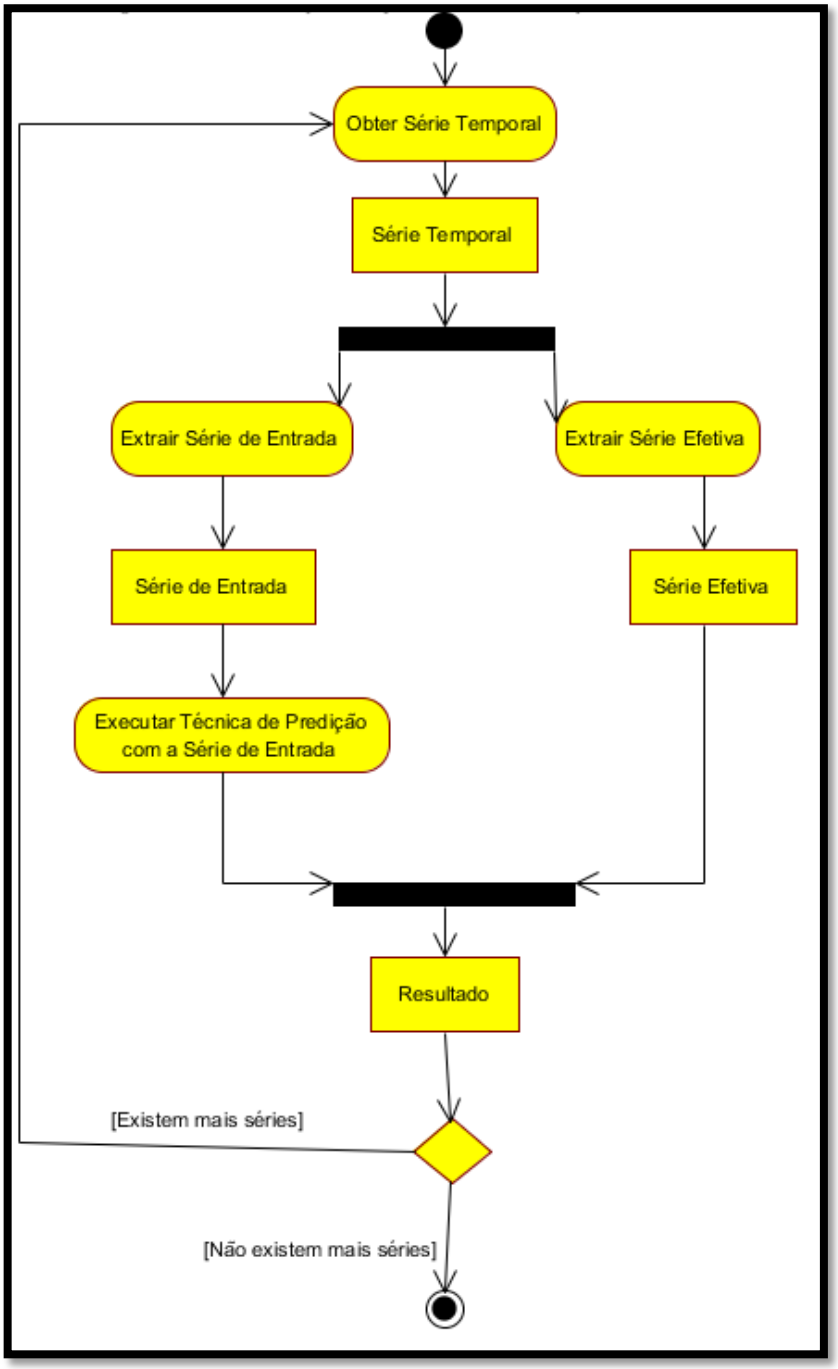

Figura 23. Fluxo de execução de um experimento.

O componente *Controlador de Entrada* utiliza a biblioteca Oceano, desenvolvida pelo Grupo de Evolução e Manutenção de Software[13] (GEMS) da Universidade Federal Fluminense. O Oceano fornece, dentre outras funcionalidades, a extração de medidas, compilação de projetos *Maven* e operações básicas em repositórios de gerência de configuração, como *checkout* e listagem de revisões, as quais foram essenciais a este trabalho.

[13] http://gems.ic.uff.br

Através de bibliotecas de medição de software, como JavaNCSS[14], NDepend[15], BCEL[16] e Javaparser[17], o Oceano disponibiliza uma interface para a implementação de extratores que lidam com o código-fonte e com arquivos binários para obter as medidas *LinesOfCode*, *CyclomaticComplexity*, *NumberOfMethods*, *NumberOfAttributes*, *LackOfCohesionInMethods*, entre outras.

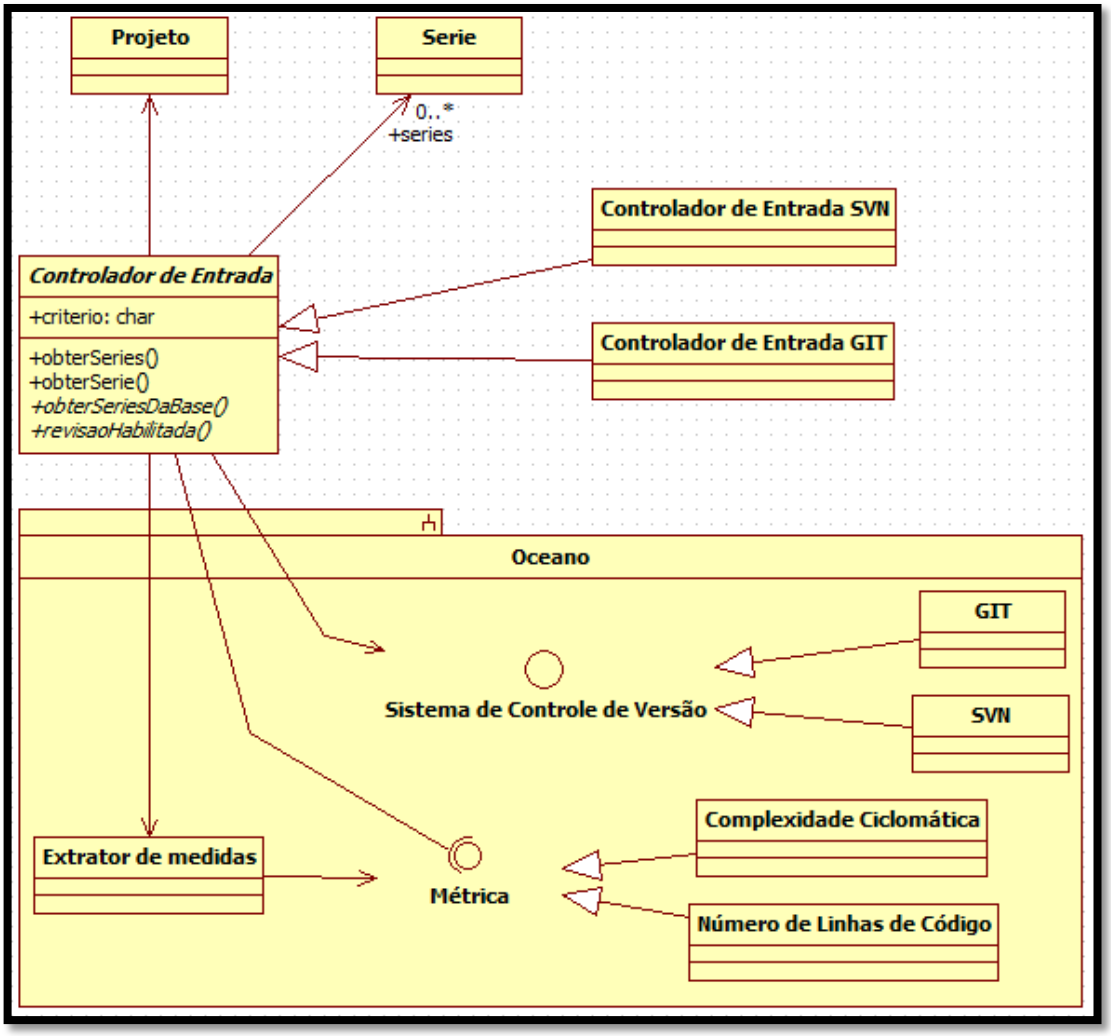

Figura 24. ControladorDeEntrada.

O Oceano, em sua versão atual, implementa entidades que realizam operações em repositórios controlados pelos sistemas de controle de versões *GIT* e *SVN*. Essas entidades provêem, no presente, as funções básicas de *checkout* e listagem de revisões e utilizam bibliotecas de manipulação a esses repositórios, como *JGit*[18] e *SVNKit*[19].

[14] http://www.kclee.de/clemens/java/javancss/
[15] http://www.ndepend.com/
[16] http://commons.apache.org/proper/commons-bcel/
[17] http://code.google.com/p/javaparser/
[18] http://www.eclipse.org/jgit/
[19] http://svnkit.com/

Para obter séries compostas de medidas do software, realiza-se um percurso pelo conjunto de revisões registradas em um repositório. Para cada revisão encontrada, realiza-se seu *checkout* e se obtém a lista de classes modificadas por ela. Cada classe dessa lista é medida (isto é, a classe e seus atributos, como complexidade, tamanho, etc.) por um dos algoritmos de medição. Esse processo se repete enquanto houver revisão a analisar e, ao final, estão formadas as sequências de medidas (isto é, as séries temporais). Esse percurso e os *checkouts* são realizados pelo método *obterSeriesDaBase*. A Figura 25 mostra o histórico de 3 classes ao longo de 5 *commits* realizados em determinado repositório, na qual cada ponto representa a ocorrência de modificação de conteúdo sucedida em cada uma. O percurso pelas revisões obtém o primeiro *commit*, no qual todas as 3 classes foram modificadas. Assim, as séries A, B e C recebem sua primeira medida, extraída com base no estado atual de cada classe. Assim continua o processo até que todos os *commits* tenham sido analisados. É importante notar que há *commits* em que uma classe não é modificada. Nesse caso, o processo simplesmente não realiza a medição nela e contempla as demais. A Figura 26 ilustra o processo descrito neste parágrafo.

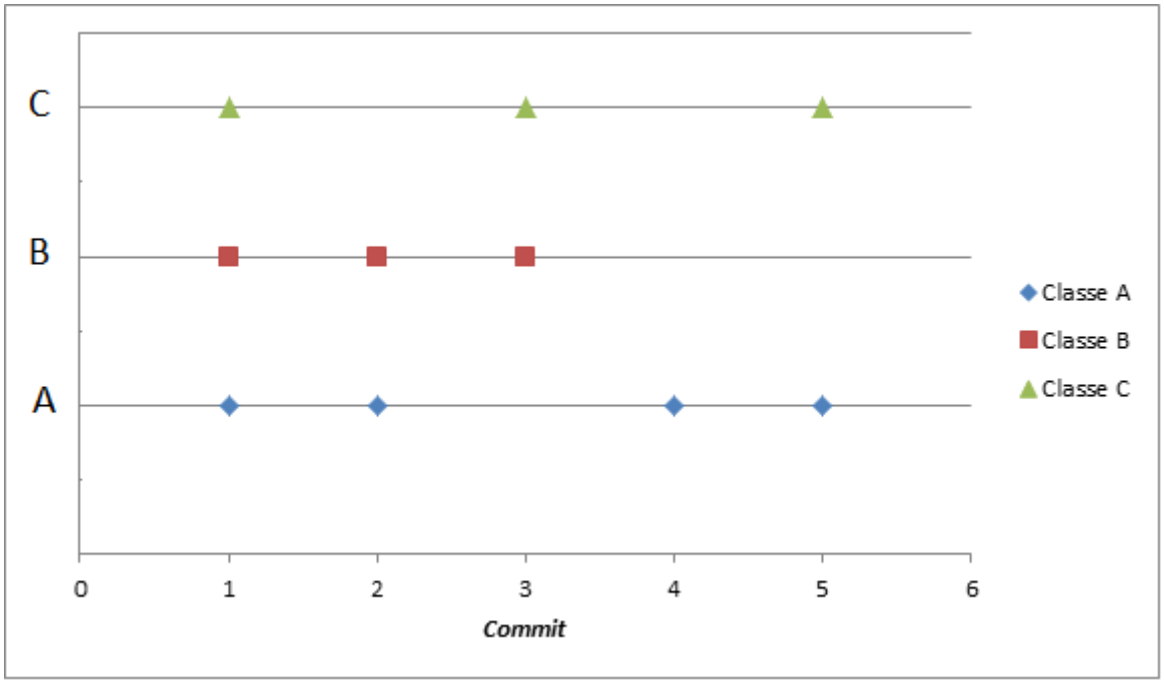

Figura 25. *Commits* armazenados em um repositório.

Durante a pesquisa, surgiu a demanda de aplicar restrições às revisões a serem medidas. São restrições como: somente aceitar revisões registradas antes/após determinada data ou antes/após determinado número de revisão. Elas são aplicadas no fluxo de execução mostrado pela Figura 26, indicadas pelas condições de guarda Revisão Habilitada e Revisão Não Habilitada. Durante o percurso pelas revisões, através do método *revisãoHabilitada*,

aquelas que não atendem a um determinado critério de aceitação, definido pelo usuário através do atributo *critério*, são desconsideradas. Na versão atual do framework, são dois os critérios: Intervalo de Datas e Intervalo de Revisões. O critério Intervalo de Datas determina que sejam aceitas apenas as revisões compreendidas entre uma data inicial e uma data final. Essas datas são definidas em termos de dia, mês, ano, hora, minuto e segundo. Assim, quando o processo iterativo obtém uma nova revisão para analisar, são verificadas tais informações temporais. Caso essas informações não atendam ao critério, a revisão não é utilizada. O critério Intervalo de Revisões tem significado similar ao anterior, no entanto se analisa se o número da revisão está dentro do intervalo compreendido entre uma revisão inicial e uma revisão final.

Ao final do fluxo da Figura 26, as séries estão prontas para serem utilizadas. Através do método *obterSeries*, elas podem ser retornadas em uma lista. Esse método pode, ainda, filtrar as séries que retorna de acordo com dois critérios: o tamanho das séries, verificando se são maiores do que um tamanho mínimo, ou a existência de revisões cujo tipo de alteração realizada seja a remoção do artefato. É possível também, através do método *obterSerie*, obter a série correspondente a um determinado artefato.

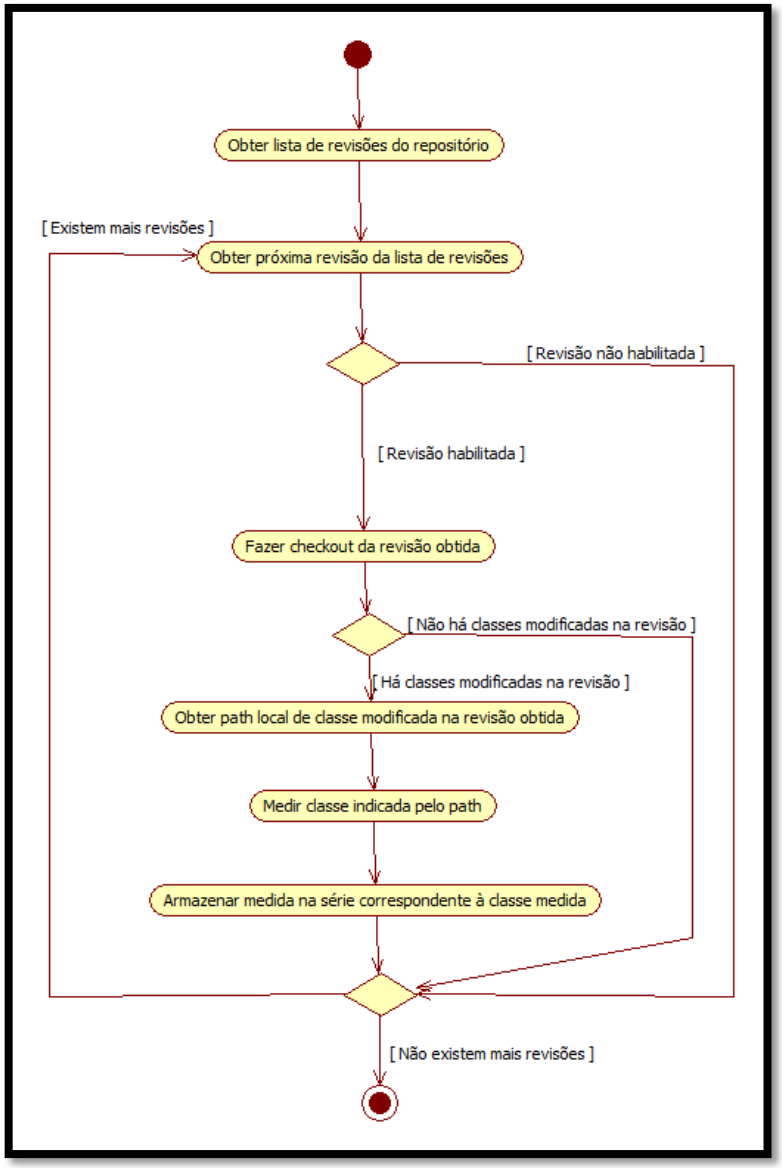

Figura 26. Fluxo de execução do processo de obtenção de medidas.

4.4 CONTROLADOR DE SAÍDA

O componente *Controlador de Saída* oferece formas diversificadas de visualizar os resultados das predições. Ele permite a visualização dos resultados por relatórios, gráficos de séries e histogramas.

Uma das funcionalidades desse componente é a transformação dos dados oriundos de experimentos, que estão na forma textual, em formatos que visam auxiliar na interpretação de seus resultados. Ele oferece métodos que buscam dados oriundos de experimentos através do Gerenciador de Experimentos (apresentado na Seção 4.5) e os exibem em forma de texto. Um deles retorna uma *String* formatada (Figura 27), contendo informações como o identificador do experimento, a técnica executada, nomes do projeto, métrica e artefato, a série de entrada,

a série de valores preditos, a série de valores efetivos e as medidas de avaliação de desempenho de cada predição, como será discutido no Capítulo 5. O outro gera um arquivo *CSV* (*Comma-Separate Values*), contendo as mesmas informações apresentadas pelo primeiro, como pode ser visto na Figura 28. Um terceiro método gera uma sumarização dos vários resultados de um experimento, através da métrica de desempenho *Erro Relativo Médio* (também discutido no Capítulo 5), possibilitando a classificação de técnicas de predição em razão da proximidade (em média) de seus valores preditos com os valores reais. Essa sumarização, assim como informações como os nomes do projeto e da métrica, quantidade de séries utilizadas no experimento e desvio-padrão da média dos erros relativos, também é salva em um arquivo *CSV* (Figura 29).

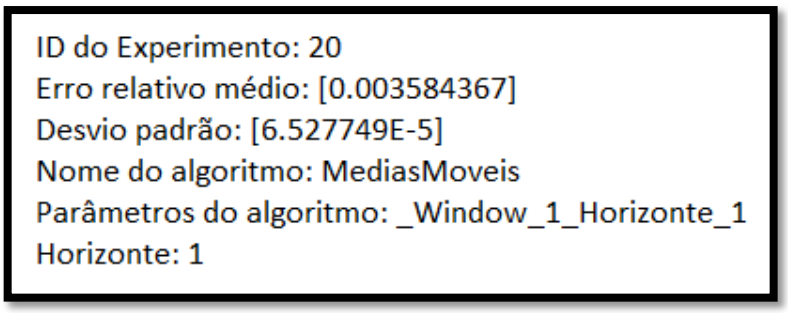

Figura 27. Exemplo de saída textual com dados do experimento.

O *Controlador de Saída* gera, além de informações textuais, gráficos de séries temporais e histogramas para auxílio nas análises dos resultados dos experimentos. Para isso, utiliza o framework gráfico *JFreeChart*[20]. Todas essas representações visuais são armazenadas em arquivos de imagem, no formato JPEG, nas dimensões 1024 pixels x 1024 pixels, em um diretório especificado pelo usuário.

ID Exp	Algoritmo	Parâmetros	Projeto	Artefato	Métrica	Entrada	Predito	Efetivo	Erro Absoluto	Erro Relativo
20	MediasMoveis	_Window_1_Horizonte_1	Apache_Ant	Artefato1	Complexidade Ciclomática	[a,b,c,d]	3,98	3,98	0	0
20	MediasMoveis	_Window_1_Horizonte_1	Apache_Ant	Artefato2	Complexidade Ciclomática	[a,b,c,d]	3,16	3,14	0,02	0,01
20	MediasMoveis	_Window_1_Horizonte_1	Apache_Ant	Artefato3	Complexidade Ciclomática	[a,b,c,d]	5,03	5,03	0	0
20	MediasMoveis	_Window_1_Horizonte_1	Apache_Ant	Artefato4	Complexidade Ciclomática	[a,b,c,d]	2,03	2,03	0	0
20	MediasMoveis	_Window_1_Horizonte_1	Apache_Ant	Artefato5	Complexidade Ciclomática	[a,b,c,d]	4,55	4,55	0	0
20	MediasMoveis	_Window_1_Horizonte_1	Apache_Ant	Artefato6	Complexidade Ciclomática	[a,b,c,d]	6,52	6,52	0	0
20	MediasMoveis	_Window_1_Horizonte_1	Apache_Ant	Artefato7	Complexidade Ciclomática	[a,b,c,d]	3,61	3,61	0	0
20	MediasMoveis	_Window_1_Horizonte_1	Apache_Ant	Artefato8	Complexidade Ciclomática	[a,b,c,d]	3,94	3,94	0	0
20	MediasMoveis	_Window_1_Horizonte_1	Apache_Ant	Artefato9	Complexidade Ciclomática	[a,b,c,d]	2,7	2,7	0	0
20	MediasMoveis	_Window_1_Horizonte_1	Apache_Ant	Artefato10	Complexidade Ciclomática	[a,b,c,d]	3,88	3,88	0	0
20	MediasMoveis	_Window_1_Horizonte_1	Apache_Ant	Artefato11	Complexidade Ciclomática	[a,b,c,d]	3,86	3,86	0	0
20	MediasMoveis	_Window_1_Horizonte_1	Apache_Ant	Artefato12	Complexidade Ciclomática	[a,b,c,d]	2,44	2,44	0	0
20	MediasMoveis	_Window_1_Horizonte_1	Apache_Ant	Artefato13	Complexidade Ciclomática	[a,b,c,d]	2,32	2,32	0	0
20	MediasMoveis	_Window_1_Horizonte_1	Apache_Ant	Artefato14	Complexidade Ciclomática	[a,b,c,d]	2,44	2,44	0	0
20	MediasMoveis	_Window_1_Horizonte_1	Apache_Ant	Artefato15	Complexidade Ciclomática	[a,b,c,d]	2,76	2,76	0	0
20	MediasMoveis	_Window_1_Horizonte_1	Apache_Ant	Artefato16	Complexidade Ciclomática	[a,b,c,d]	3,71	3,71	0	0
20	MediasMoveis	_Window_1_Horizonte_1	Apache_Ant	Artefato17	Complexidade Ciclomática	[a,b,c,d]	2,88	2,88	0	0
20	MediasMoveis	_Window_1_Horizonte_1	Apache_Ant	Artefato18	Complexidade Ciclomática	[a,b,c,d]	3,43	3,43	0	0
20	MediasMoveis	_Window_1_Horizonte_1	Apache_Ant	Artefato19	Complexidade Ciclomática	[a,b,c,d]	3,07	3,07	0	0
20	MediasMoveis	_Window_1_Horizonte_1	Apache_Ant	Artefato20	Complexidade Ciclomática	[a,b,c,d]	2,1	2,06	0,04	0,02

Figura 28. Exemplo de arquivo *CSV* contendo os resultados individuais da predição de cada série.

[20] http://www.jfree.org/jfreechart/

ID Exp	Algoritmo	Parâmetros	Qtd de Séries	Erro Relativo Médio 1	Desvio Padrão 1
20	MediasMoveis	_Window_1_Horizonte_1	50	0,004	6,53E-005
21	KNN	_K_1_MetDist_euclideana_medidaTendCentral_mediana_horizonte_1	50	0,106	0,01
22	RegressaoLine	_K_1_MetDist_euclideana_medidaTendCentral_mediana_horizonte_1	50	0,025	0,02
23	RegressaoQua	_Horizonte_1	50	0,105	0,01
24	MediasMoveis	_Horizonte_1	50	0,147	0,02

Figura 29. Exemplo de arquivo *CSV* contendo os dados do experimento.

O gráfico de série é a representação de um conjunto de coordenadas relativas aos valores de uma série em um plano cartesiano. Na implementação atual do Peixe-Dipnoico, o eixo Y desse plano representa os valores das medidas das séries, enquanto o eixo X representa o identificador ordinal (isto é, o número 1 para o primeiro *commit*, 2 para o segundo, e assim sucessivamente) do *commit* que gerou o valor do eixo Y. O *Controlador de Saída* oferece ao usuário a construção de gráficos de séries. Transmite-se a esse método uma lista de séries que serão exibidas no arquivo de imagem gerado. Dessa forma, é possível plotar mais de uma série no mesmo gráfico, possibilitando comparações. A Figura 30 mostra um exemplo de gráfico de série gerado pelo *framework*. Nessa figura, é mostrada uma série que representa a evolução dos valores da métrica Complexidade Ciclomática ao longo de 53 *commits*.

Os histogramas representam em um gráfico de barras a contagem de valores (ou intervalos de valores) assumidos por determinada variável dentro de uma amostra experimental. No eixo X, são definidos que valores ou intervalos serão contabilizados, enquanto o eixo Y mostra sua contagem (ou frequência). Os histogramas são gerados com base em três variáveis, obtidas dos resultados dos experimentos: erro absoluto, erro relativo e tamanho das séries. A Figura 31 mostra um exemplo de histograma gerado pela abordagem. Nessa figura, é mostrado um histograma que representa a contagem do número de séries em um experimento contida em cada faixa de tamanho.

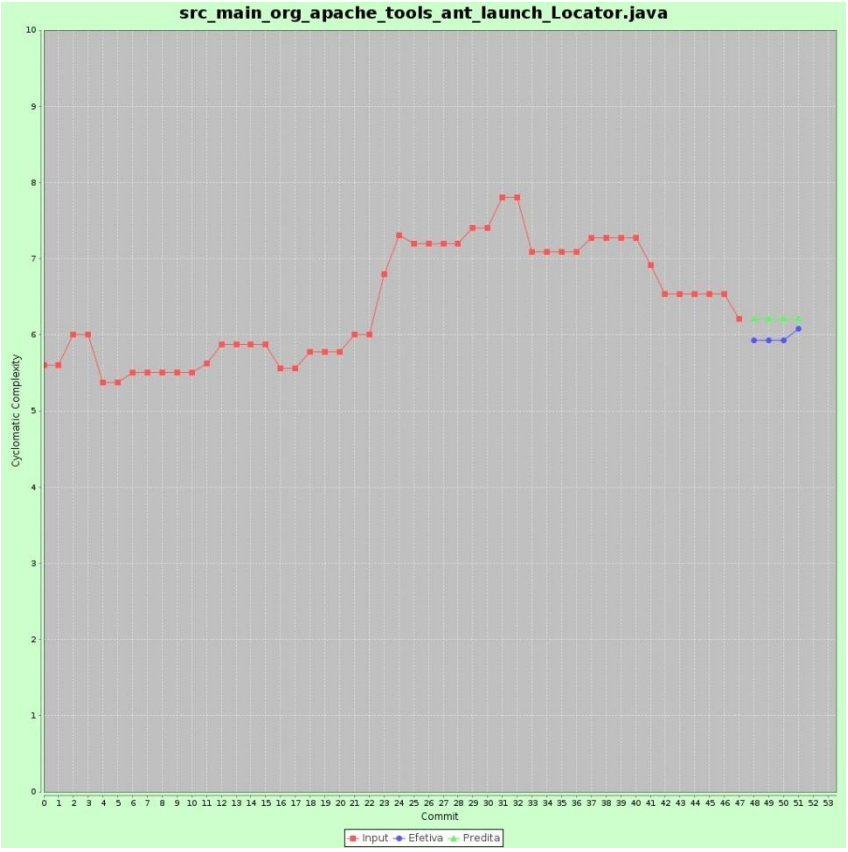

Figura 30. Gráfico de série gerado pela abordagem.

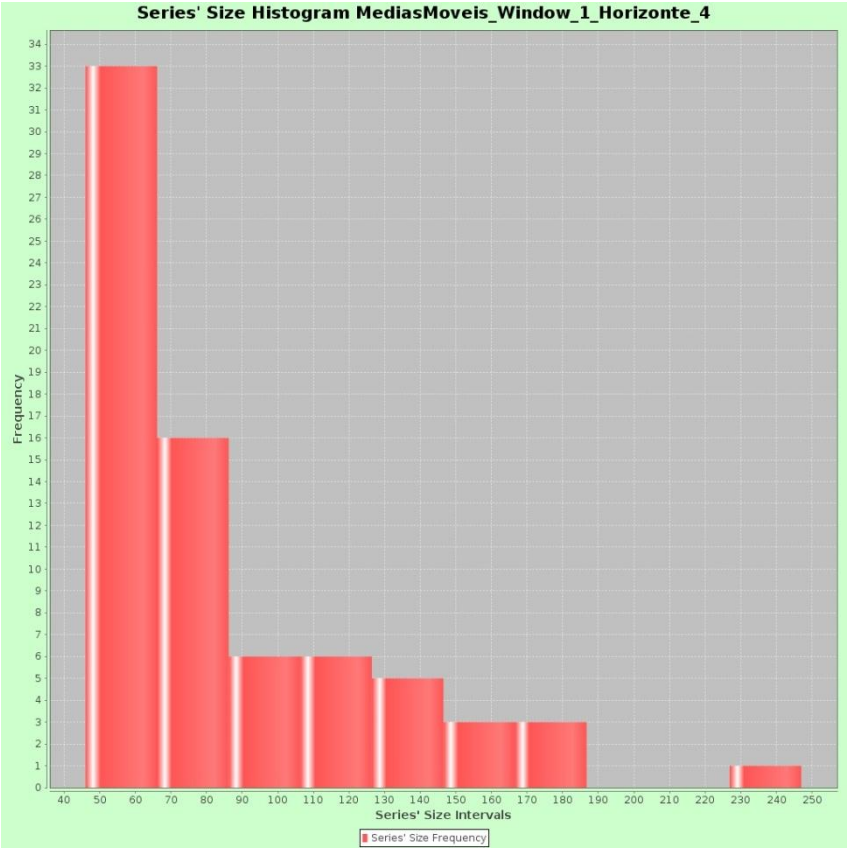

Figura 31. Histograma gerado pela abordagem.

4.5 KERNEL

O componente *Kernel* é o centro do funcionamento do *framework*. O *Kernel* possui dois modos de operação: um de experimentação e outro de uso.

No modo de experimentação, o componente conta com a entidade *GerenciadorDeExperimentos*, a qual se responsabiliza pelo controle da execução dos experimentos, gerenciando-os desde sua criação até a sua apresentação ao usuário. Para cumprir esse objetivo, ela se associa com as entidades *ControladorDeEntrada* e *ControladorDeSaida*, relacionadas, respectivamente, aos componentes *Controlador de Entrada* e *Controlador de Saída*, como mostrado pela Figura 32. Como pode ser observado nessa figura, *GerenciadorDeExperimentos* é capaz de gerenciar múltiplas fontes de dados ao mesmo tempo, possibilitando, por exemplo, a coexistência de experimentos com séries de diferentes projetos.

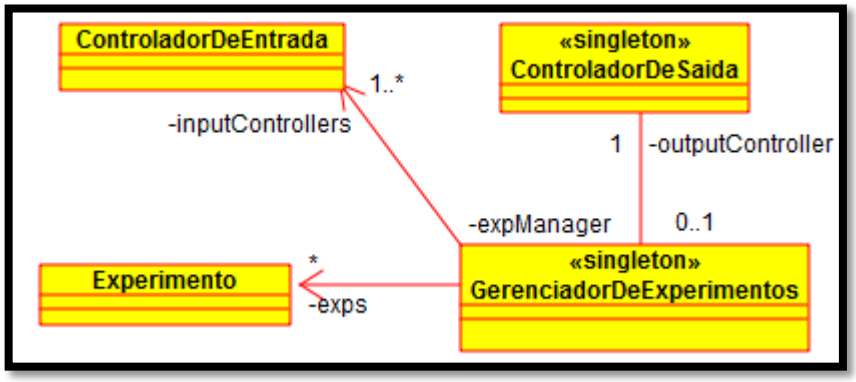

Figura 32. Entidades responsáveis pela execução de experimentos.

As entidades mostradas na Figura 32 constituem o fluxo do processo de execução de experimentos. Esse fluxo, mostrado na Figura 33, transcorre da seguinte forma:

1. Adicionar um Controlador de Entrada. Outros podem ser adicionados, conforme a necessidade.
2. Escolher um dos Controladores adicionados, obtendo dele as séries temporais a serem utilizadas.
3. Adicionar um experimento, definindo seu conjunto de séries (obtido no Passo 2) e sua técnica de predição (ver Seção 4.2). Outros experimentos podem ser adicionados, conforme a necessidade.
4. Escolher um dos experimentos adicionados, executando-o em seguida.
5. Após a execução do experimento, obter os resultados através de uma (ou mais) das formas de visualização, apresentadas na Seção 4.4.

Já no modo de uso, o *framework* é utilizado por aplicações que não estejam ligadas à realização de experimentos com técnicas de predição, portanto não é necessária a existência do orquestrador *GerenciadorDeExperimentos*. Dessa forma, o *Kernel*, nesse modo de uso, relaciona os componentes do *framework*. Ele inicialmente acessa o *Metamodelo* para obtenção das séries temporais geradas pelo *Controlador De Entrada*. Em seguida, utiliza o componente *Técnica de Predição* para a realização da predição, encaminhando os resultados ao *Controlador De Saída* ou simplesmente apresentando os valores preditos da série usada como entrada.

Entretanto, quando o *framework* é utilizado por aplicações que não estejam ligadas à realização de experimentos com técnicas de predição, não é necessária a existência do orquestrador *GerenciadorDeExperimentos*. Dessa forma, o *Kernel* torna-se puramente conceitual, relacionando os componentes do *framework*. Ele inicialmente acessa o *Metamodelo* para obtenção das séries temporais geradas pelo *Controlador De Entrada*. Em seguida, utiliza o componente *Técnica de Predição* para a realização da predição, encaminhando os resultados ao *Controlador De Saída* ou simplesmente apresentando os valores preditos da série usada como entrada.

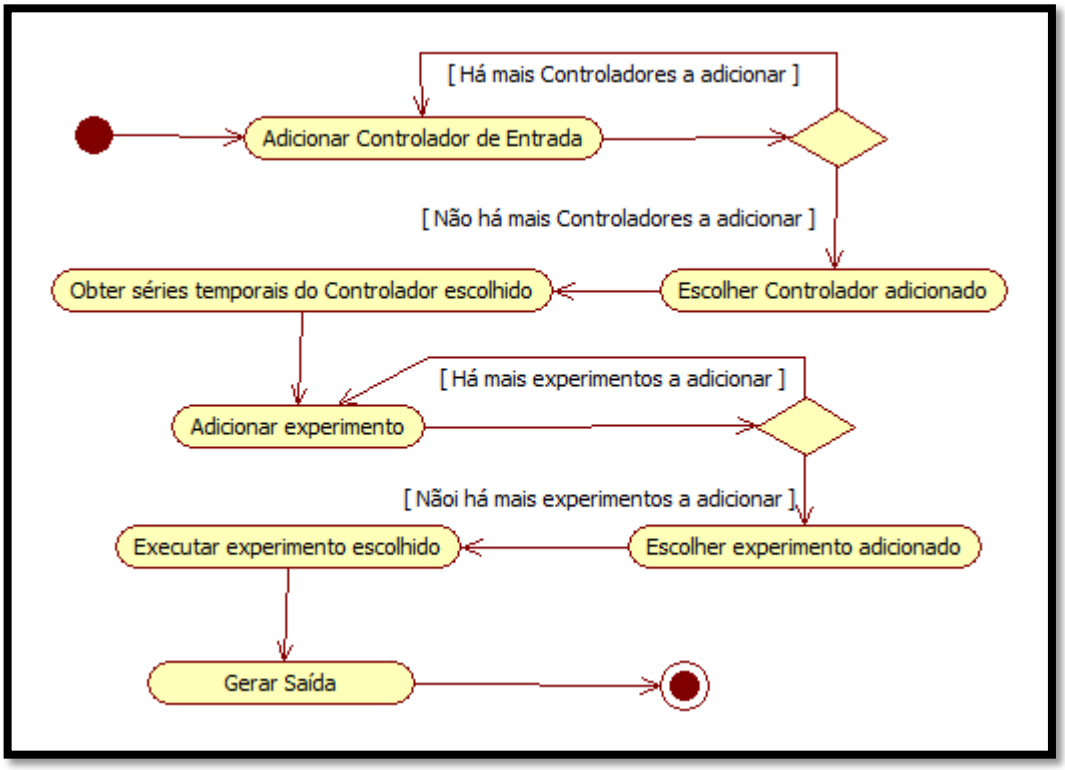

Figura 33. Fluxo do processo de execução dos experimentos.

4.6 TÉCNICA DE PREDIÇÃO

Como discorrido anteriormente, o componente *Técnica de Predição* (Figura 18) é um ponto de extensão do *framework*. Esse componente é um arcabouço para a modelagem de técnicas de predição, possibilitando que sejam incorporados às aplicações previamente definidas que fazem uso do *framework* sem a necessidade de realizar modificações na estrutura deste último. *Técnica de Predição* é constituído pelas entidades mostradas pela Figura 34.

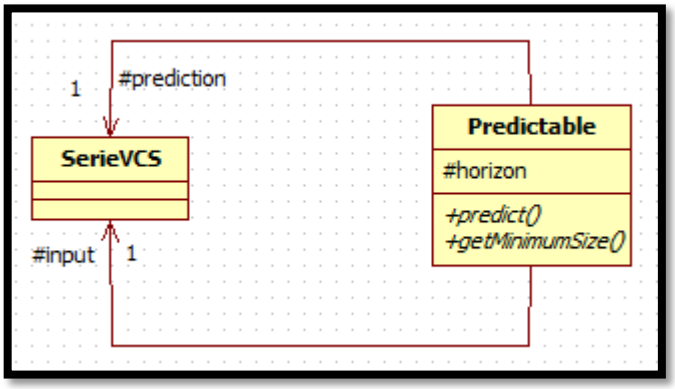

Figura 34. Estrutura do componente *Técnica de Predição*.

A entidade abstrata *Predictable* define o *horizonte* de predição (Capítulo 2). A partir de uma série de entrada (*entrada*), uma técnica de predição baseada nessa entidade deve realizar um processamento no método *predizer* e, então, gerar os valores preditos, que são armazenados em *prediction*. Essa técnica deve especificar, através do método *getTamanhoMínimo*, o tamanho mínimo de série com o qual consegue operar.

A versão atual do Peixe-Dipnoico conta com quatro tipos de técnica de predição, que se baseiam em *Predictable*: (1) *KNN*, (2) *Ajuste de Funções*, (3) *Autorregressão* e (4) *Dinâmico*. Nas subseções seguintes, todos esses tipos de técnica mencionados e as técnicas que deles derivam serão apresentados em detalhes. É importante ressaltar que os nomes utilizados para as técnicas e seus tipos implementados neste trabalho não necessariamente são encontrados na literatura, mas fazem referência às suas particularidades, descritas nos respectivos parágrafos.

4.6.1 KNN

As técnicas do tipo *KNN* são inspiradas em uma técnica homônima pertencente à área de aprendizagem de máquina, como mostrado no Capítulo 2. Sua principal premissa é a de que um comportamento passado, observado na variação dos valores de uma ou mais séries,

tende a se repetir no futuro para séries similares. Elas necessitam de uma base de séries temporais, onde procuram as *k* séries mais similares à série de entrada, considerando seus valores absolutos, para compor a predição.

Como dito anteriormente, as técnicas do tipo *KNN* demandam que sejam encontradas séries similares entre si. Em se tratando de séries, uma forma de calcular essa similaridade é através do cálculo de distâncias. Em outras palavras, quanto menor a distância entre duas séries, maior a similaridade entre elas, e vice-versa. Essa abordagem foi utilizada em FERRERO (2009), como mostrado no Capítulo 2. A similaridade é calculada como a distância entre duas séries. As métricas de distância utilizadas são baseadas nas tradicionais Distância Euclideana e Distância de *Manhatann*, usadas no cálculo de distâncias vetoriais. Nesse cálculo, como explicado no Capítulo 2, os pontos da sequência de valores são considerados pontos no espaço n-dimensional, sendo tratados dessa forma pelas métricas. Para duas séries $S_1 = \{x_1, x_2, ..., x_n\}$ e $S_2 = \{y_1, y_2, ..., y_n\}$, definem-se a Distância Euclideana e a Distância de *Manhatann*, como mostrado, respectivamente, por **(5)** e por **(6)**.

$$D(S_1, S_2) = \sqrt[2]{(x_1 - y_1)^2 + (x_2 - y_2)^2 + ... + (x_n - y_n)^2} \quad \textbf{(5)}$$

$$D(S_1, S_2) = |x_1 - y_1| + |x_2 - y_2| + ... + |x_3 - y_3| \quad \textbf{(6)}$$

Definido o conceito de similaridade, central ao *KNN*, é apresentado a seguir o fluxo de execução da técnica. Cada passo será explicado em detalhes nos parágrafos seguintes.

1. Medir a similaridade (usando uma das métricas de distância disponíveis) entre a série de entrada e cada subsérie de cada série da base. Neste trabalho, uma subsérie é o resultado do recorte de uma série, o qual começa em determinada posição e termina em outra.
2. Procurar as *k* subséries cujas similaridades com a série de entrada, calculadas no Passo 1, tenham sido as menores.
3. Obter as subséries que comporão a predição, compreendidas do primeiro elemento após cada subsérie escolhida como a mais similar até o último elemento da série correspondente.
4. Igualar os tamanhos das subséries escolhidas para a predição (geradas no Passo 3).
5. Combinar elementos em posições correspondentes usando **média aritmética** ou **mediana**.

No **Passo 1**, percorrem-se as subséries de cada série da base, calculando para cada uma sua similaridade com relação à série de entrada. A similaridade é armazenada em uma estrutura de dados própria para essa finalidade, assim como as outras informações que

localizam cada subsérie, isto é, a identificação da subsequência e de sua sequência relacionada (doravante chamada de *InfoSimilaridade*).

Cada série comparada à entrada tem um número máximo de subséries que dela pode ser obtido. Esse número é calculado com base no tamanho da série, já que toda subsérie deve ter o mesmo tamanho que a entrada. O número máximo *MAX* de subséries será dado por **(7)**. Por exemplo, se a entrada tem dois elementos e uma série da base tem três elementos ($S = \{s_1, s_2, s_3\}$), é possível ter no máximo 3 – 2 + 1 subséries, as quais serão: $\{s_1, s_2\}$ e $\{s_2, s_3\}$.

$$MAX(s) = |s| - |entrada| + 1 \quad \textbf{(7)}$$

Nem todas as subséries podem ser utilizadas no processamento, pois é preciso sobrar elementos para compor a predição, de acordo com o horizonte. Portanto, as últimas subséries sempre serão desconsideradas. Utilizando o exemplo anterior, se o horizonte é 1, então é preciso sobrar pelo menos um elemento para compor a predição. Nesse caso, só será possível utilizar uma subsérie das duas possíveis. Para cada série comparada à entrada, o número máximo *N* de subséries que poderão ser utilizadas no cálculo de similaridade com a entrada é dado por **(8)**.

$$N(s) = MAX(s) - horizonte \quad \textbf{(8)}$$

O **Passo 1** considera três possibilidades quanto ao tamanho da série da base para o cálculo das similaridades de suas subséries com a entrada:

- Se o tamanho da série da base é menor que o horizonte, ela não será utilizada, pois não sobrará elemento para compor a predição.
- Caso contrário, se o número de elementos que sobram (dada a **(8)**) é maior ou igual à entrada, então a entrada inteira poderá ser utilizada no cálculo de similaridades (pois se garante que sobrará o número suficiente de elementos).
- Caso contrário, a entrada será podada nos seus elementos de menor posição (mais antigos) até possuir um tamanho que possibilite a sobra de elementos para compor a predição.

Para facilitar a compreensão, será apresentado a seguir como exemplo o resultado do processamento do **Passo 1**. A Figura 35 contém séries em uma base fictícia. Considerando a descrição do **Passo 1** nos parágrafos anteriores e supondo que o horizonte de predição seja 1, seria criada a tabela de similaridade mostrada na Tabela 2. Na tabela, a coluna *Entrada* indica a série da Figura 35 utilizada como entrada do *KNN*. A coluna *Entrada Utilizada* mostra o resultado da poda realizada na entrada, caso tenha acontecido, ou, caso não tenha acontecido a poda, mostra a série em sua totalidade, conforme indicada pela coluna *Entrada*. A coluna *ID* é

a identificação da subsérie indicada pela coluna *Subsérie*. Por fim, a coluna *Distância* indica a distância entre a *Entrada Utilizada* e a *Subsérie*, utilizando a Distância de *Manhatann*.

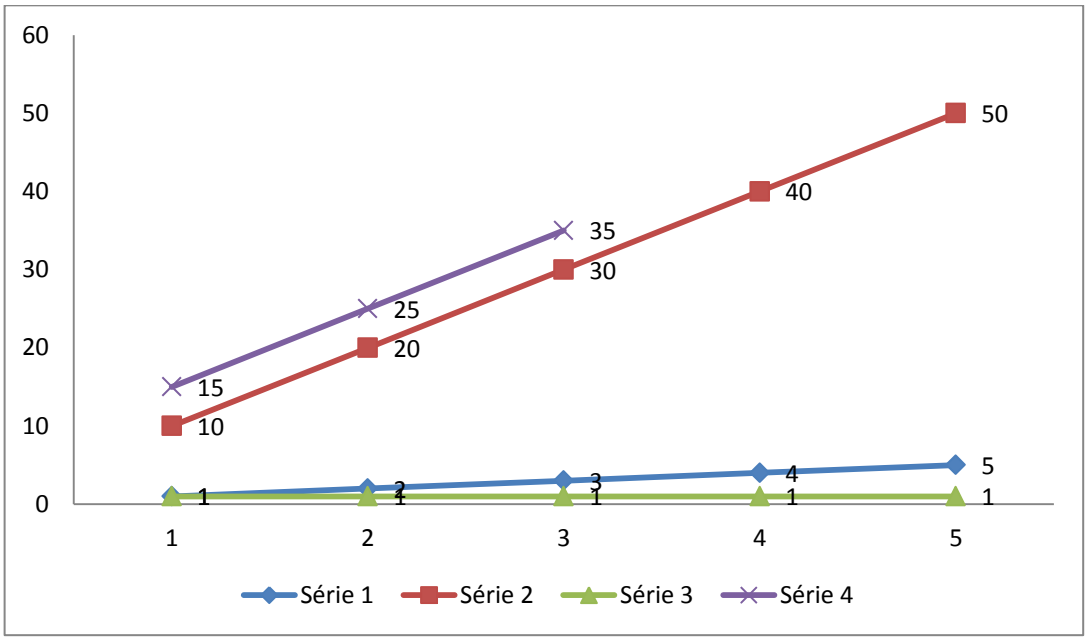

Figura 35. Conjunto fictício de séries.

Tabela 2. Distâncias entre as séries da Figura 35.

Entrada	Entrada utilizada	ID	Subsérie	Distância
Série 1	{2,3,4,5}	1	{10,20,30,40}	86
	{2,3,4,5}	2	{1,1,1,1}	10
	{4,5}	3	{15,25}	31
Série 4	{15,25,35}	4	{1,2,3}	69
	{15,25,35}	5	{2,3,4}	66
	{15,25,35}	6	{10,20,30}	15
	{15,25,35}	7	{20,30,40}	15
	{15,25,35}	8	{1,1,1}	72
	{15,25,35}	9	{1,1,1}	72

É importante observar que a poda de elementos da entrada gera o efeito de privilegiar séries com tamanhos menores na escolha das mais similares. Isso acontece porque as fórmulas, como pode ser visto em **(5)** e em **(6)**, são, em essência, um somatório das diferenças entre os pontos. Dessa forma, séries maiores tendem a gerar valores de distância maiores. Esse efeito pode ser observado no exemplo da Tabela 2, onde a subsérie 4 tem preferência sobre a subsérie 2, sendo perceptível que essa última está mais perto. Para lidar com esse

problema, os valores das distâncias foram ponderados pelo tamanho da série (de entrada) podada, gerando um valor que pode ser considerado como a distância média entre cada par de elementos das séries envolvidas. Após esse ajuste nas distâncias, os valores de similaridade na Tabela 2 foram convertidos para aqueles apresentados na Tabela 3. No exemplo, por se tratar de séries muito pequenas e com tamanhos muito próximos, o ajuste nas distâncias não eliminou esse efeito. É possível, entretanto, pensar em uma simples situação em que exista uma série de entrada S_E = {1,1,1,...,1}, com 1000 elementos, uma série S_1 = {10,10,10,...,10}, com 100 elementos, e uma série S_2 = {2,2,2,...,2}, com 1000 elementos. Realizando o processo de poda originalmente proposto, a distância (de *Manhatann*) entre S_E e S_1 seria 900 e a distância entre S_E e S_2 seria 1000, o que é perceptivelmente contraditório. Ajustando as distâncias, os valores seriam 9, no primeiro caso, e 1, no segundo, o que seria mais plausível, haja vista os valores das séries envolvidas.

Tabela 3. Distâncias entre as séries da Figura 35, ponderadas pelo tamanho da série de entrada após a poda.

Entrada	Entrada utilizada	ID	Subsérie	Distância
Série 1	{2,3,4,5}	1	{10,20,30,40}	21,5
	{2,3,4,5}	2	{1,1,1,1}	2,5
	{4,5}	3	{15,25}	15,5
Série 4	{15,25,35}	4	{1,2,3}	23
	{15,25,35}	5	{2,3,4}	22
	{15,25,35}	6	{10,20,30}	5
	{15,25,35}	7	{20,30,40}	5
	{15,25,35}	8	{1,1,1}	24
	{15,25,35}	9	{1,1,1}	24

No **Passo 2**, procura-se na lista de *InfoSimilaridades* geradas no **Passo 1** referências às *k* séries mais similares. Ao final desse passo, é formada uma lista contendo tais referências, sendo cada uma relacionada a uma série distinta. Em outras palavras, mesmo que haja na mesma série mais de uma subsérie apontada como as mais similares, somente a primeira encontrada (necessariamente a menor dentro da série) será a utilizada. Considerando as séries da Figura 35, para *k* com valor igual a 2, seriam selecionadas da Tabela 2 as subséries 2 e 3, caso a entrada fosse a Série 1, e as subséries 6 e 5, caso a entrada fosse a Série 4.

No **Passo 3**, as referências às *k* subséries mais similares (do **Passo 2**) são utilizadas para recuperar as séries das quais as subséries fazem parte. É retirado de cada série recuperada

o fragmento que corresponda à continuação da subsérie mais similar, isto é, a subsérie que vai do primeiro elemento após o último da subsérie mais similar até o final da série. Considerando as séries da Figura 35, para o caso em que a entrada é a Série 1, são obtidos os fragmentos {1} e {35}, e para o caso em que a entrada é a Série 4, são obtidos os fragmentos {40,50} e {5}.

No **Passo 4**, obtém-se o menor tamanho de série dentre aquelas encontradas no **Passo 3**. Em seguida, as séries são recortadas, sendo consideradas somente as posições que vão de 0 (zero) até a posição correspondente a esse menor tamanho encontrado. Considerando as séries da Figura 35, os fragmentos obtidos no **Passo 3** tornam-se {1} e {35}, quando a entrada é a Série 1, e {40} e {5}, quando a entrada é a Série 4.

Por fim, no **Passo 5** é obtida a média ou a mediana dos elementos situados em posições correspondentes nos fragmentos obtidos no **Passo 4**, isto é, a média ou a mediana de todos os elementos situados na primeira posição de cada um dos fragmentos é calculada, gerando o primeiro valor predito, e assim por diante. Também são calculados os desvios-padrões referentes ao cálculo das médias ou medianas. Considerando as séries da Figura 35 e usando a média, para o caso em que a entrada é a Série 1, o valor predito é 18, e para o caso em que a entrada é a Série 4, o valor predito é 22,5.

Durante o desenvolvimento do *KNN*, observando os erros de suas predições, percebeu-se que a utilização direta dos valores originais das séries não trazia bom desempenho às predições. Esses resultados levaram à hipótese de que formas alternativas de representação poderiam melhorar os resultados. Para lidar com esse problema, foram criadas duas abordagens que implementam essas formas alternativas, chamadas de **cálculo de intervalos relativos** e **cálculo de intervalos absolutos**. Ambas são explicadas no parágrafo seguinte.

Em ambas as abordagens citadas anteriormente, é necessário calcular a diferença entre cada elemento da série e o seu antecessor, começando pelo segundo. Essa diferença entre elementos é chamada de **intervalo**. O primeiro método divide cada diferença pelo delta, calculado como a diferença entre o maior e o menor valor encontrado na série. Por dividir pelo valor do delta, cada intervalo calculado é relativo à amplitude da série, sendo, portanto, chamado de **intervalo relativo**. No cálculo do intervalo relativo, se o valor de delta for zero, todos os elementos da série transformada serão zeros, indicando que não há variação entre elementos consecutivos da série original. O segundo método transforma a série utilizando diretamente essas diferenças. Por não usar o valor do delta, cada intervalo calculado é chamado de **intervalo absoluto**. É importante notar que, nos dois casos, a série transformada tem um elemento a menos em relação à original. Por exemplo, seja a série S = {1,2,3,4,5},

com cinco elementos. O primeiro método retornaria a série S’ = {(2-1)/4, (3-2)/4, (4-3)/4, (5-4)/4} = {0.25, 0.25, 0.25, 0.25}, com quatro elementos. Já o segundo método retornaria a série S’’ = {(2-1),(3-2),(4-3),(5-4)} = {1,1,1,1}, com quatro elementos. Essas transformações são utilizadas pelas técnicas KNN-IR (apresentado na Subseção 4.6.1.1) e KNN-IA (apresentado na Subseção 4.6.1.2).

4.6.1.1 KNN-IR

A técnica *KNN-IR* (KNN com Intervalos Relativos) é uma evolução do ancestral *KNN*. Ela realiza uma transformação nas séries da base e na entrada, aplicando a abordagem de **cálculo de intervalos relativos**. Assim, por exemplo, a Série 1 (Figura 35) torna-se {(2-1)/(5-1), (3-2)/(5-1), (4-3)/(5-1), (5-4)/(5-1)} = {0.25, 0.25, 0.25, 0.25}.

Essa transformação nas representações das séries traz o benefício de se detectar as séries mais similares de acordo com sua forma, em vez de ter como parâmetro a proximidade em relação a seus valores absolutos. Se a Série 1 for comparada visualmente com a Série 2, as duas, que apresentam um comportamento crescente e constante, serão consideradas similares. No entanto, considerando o funcionamento do seu ancestral, o *KNN*, a Série 3 seria mais similar à Série 1 do que a Série 2, apesar de possuir um comportamento totalmente diferente.

O funcionamento do *KNN-IR* é, em essência, semelhante ao do KNN do Passo 1 ao Passo 4, devendo-se observar que não são utilizados os valores absolutos das séries. Entretanto, o Passo 5 é modificado, pois é preciso transformar a informação no formato de intervalo relativo em valores absolutos. Dessa forma, o primeiro valor da sequência de intervalos relativos (que é percentual), retornado do Passo 3, é somado a 1 (equivalente a 100%), resultando em um novo valor que representa percentualmente o aumento ou redução (este último no caso de intervalo relativo negativo) de um valor com relação ao seu antecessor. Em seguida, essa soma é multiplicada inicialmente ao último valor da série de entrada. O mesmo processo é repetido aos demais valores da sequência de intervalos relativos com relação aos novos valores preditos.

Para ilustrar todo esse processo, sejam as séries S_1 = {1,2,3,4} e a série similar S_{sim} = {10,20,30,40,50,60}. Após o cálculo dos intervalos relativos, as séries são transformadas para S_1’ = {0.33, 0.33, 0.33} e S_{sim}’ = {0.2, 0.2, 0.2, 0.2, 0.2}. Após o Passo 3, o último valor de S_1, o valor 4, é multiplicado por 1 + 0.2 (primeiro valor da sequência de intervalos relativos), isto é, 1.2. Dessa forma, o primeiro valor predito é 4.8. O segundo valor predito é obtido com o uso do segundo valor da sequência de intervalos relativos somado com 1, sendo essa soma multiplicada ao primeiro valor predito, isto é, 4.8 * 1.2 = 5.76.

4.6.1.2 KNN-IA

A técnica *KNN-IA* (KNN com Intervalos Absolutos) também é uma evolução do ancestral *KNN* e, de modo parecido, realiza uma transformação nas séries da base e na entrada, aplicando a abordagem **cálculo de intervalos absolutos**. Assim, por exemplo, a Série 1 (Figura 35) torna-se {(2-1), (3-2), (4-3), (5-4)} = {1,1,1,1}. Assim como acontece ao *KNN-IR*, o *KNN-IA* consegue perceber similaridades quanto ao comportamento da série.

O funcionamento do *KNN-IA* é, em essência, semelhante ao do *KNN* do Passo 1 ao Passo 4, devendo-se observar que não são utilizados os valores absolutos das séries. Entretanto, o Passo 5 é modificado, pois é preciso transformar a informação no formato de intervalo absoluto em valores absolutos. Dessa forma, o primeiro valor da sequência de intervalos absolutos, retornado do Passo 3, é somado inicialmente ao último valor da série de entrada. O mesmo processo é repetido aos demais valores da sequência de intervalos absolutos com relação aos novos valores preditos.

Para ilustrar todo esse processo, sejam as séries S_1 = {1,2,3,4} e a série similar S_{sim} = {10,11,12,13,14,15}. Após o cálculo dos intervalos absolutos, as séries são transformadas para S_1' = {1, 1, 1} e S_{sim}' = {1,1,1,1,1}. Após o Passo 3, o último valor de S_1, o valor 4, é somado com 1 (primeiro valor da sequência de intervalos absolutos). Dessa forma, o primeiro valor predito é 5. O segundo valor predito é obtido com o uso do segundo valor da sequência de intervalos absolutos somado com o primeiro valor predito, isto é, 5 + 1 = 6.

4.6.2 AUTORREGRESSÃO

As técnicas do tipo *Autorregressão* tentam extrair da própria série informações que gerem predições. Essas informações são formadas inicialmente pelos valores mais recentes da série de entrada, isto é, os últimos. Os valores e a quantidade de elementos obtidos da série são definidos por uma **janela** (*window*) de determinado tamanho. Assim, supondo-se a entrada {a, b, c, d, e}, se o tamanho da janela é 3, então a janela será a subsérie {c, d, e}, correspondente aos últimos três elementos da série de entrada. Deve-se ressaltar que o tamanho da série de entrada deve ser menor ou igual ao valor de *window*, caso contrário a predição não pode ser processada.

As técnicas do tipo *Autorregressão* realizam combinações lineares entre os elementos da janela para compor uma predição, combinando-os através da aplicação de coeficientes definidos por técnicas que descendem desse tipo. Cada coeficiente é multiplicado por um elemento da janela, sendo o primeiro coeficiente combinado com o primeiro compreendido na

janela (isto é, o mais antigo), e assim por diante. Supondo que a coleção de coeficientes usada seja $\{c_1, d_1, e_1\}$, para uma entrada {a, b, c, d, e}, o valor predito **f** seria o resultado do cálculo $c_1*c + d_1*d + e_1*e$.

Os elementos da janela são atualizados a cada nova predição. Assim, a cada novo valor predito obtido, a janela o engloba, como se o valor predito passasse a fazer parte da série de entrada. Ao englobá-lo, o primeiro elemento da janela (isto é, o mais antigo) é descartado, os restantes são deslocados à esquerda (isto é, o segundo torna-se o primeiro, e assim por diante) e o valor predito é inserido como o último elemento da janela. No exemplo do parágrafo anterior, seria como considerar que a nova série de entrada passasse a ser {a,b,c,d,e,**f**}. Assim, a nova janela seria composta por {d,e,f} (o elemento c foi eliminado). Os coeficientes nunca são alterados.

4.6.2.1 MÉDIAS MÓVEIS

A técnica *Médias Móveis* é uma evolução do ancestral abstrato *Autorregressão*. A *Médias Móveis* faz a predição através da média aritmética dos elementos na janela. Todos os coeficientes são definidos em número igual ao da janela *window* e possuem o valor $1/_{window}$. Assim como acontece no funcionamento de toda técnica do tipo *AutoRegressão*, todo novo valor predito substitui o valor mais antigo da janela, entrando no cálculo da média com os restantes.

4.6.3 AJUSTE DE FUNÇÕES

As técnicas do tipo *Ajuste de Funções* procuram encontrar as constantes que formam uma curva gerada por uma função polinomial (*Polinômio*), de modo que ela descreva satisfatoriamente a série de entrada (amostra). Definida a função, ela pode ser utilizada para construir predições dos N valores após o último medido na série. Para encontrar a curva de melhor ajuste, é construído um sistema de equações baseado no objetivo de minimizar o erro quadrático (Método dos Mínimos Quadrados) (CLÁUDIO; MARINS, 1994).

No processamento da predição, realiza-se a resolução do sistema de equações para encontrar as constantes da função. Encontradas essas constantes, o polinômio está definido para qualquer valor real usado como parâmetro. Usando como parâmetros da função polinomial valores que extrapolem o tamanho da série de entrada, seu valor de retorno torna-se a predição. Por exemplo, em uma série $S = \{s_1,s_2,s_3,s_4,s_5\}$, com posições $P = \{p_1,p_2,p_3,p_4,p_5\}$, para obter o valor predito $\mathbf{s_6}$, utiliza-se como parâmetro para a função polinomial o valor $\mathbf{p_6}$.

4.6.3.1 REGRESSÃO LINEAR

A técnica *Regressão Linear* é a evolução do ancestral abstrato *Ajuste de Funções*. Na *Regressão Linear,* a curva para a qual se calcula os valores das constantes é a gerada por uma função linear, isto é, uma reta. Seu objetivo é encontrar as constantes **a** e **b** da equação de reta $f(x) = ax + b$.

4.6.3.2 REGRESSÃO QUADRÁTICA

A técnica *Regressão Quadrática* é a evolução do ancestral abstrato *Ajuste de Funções*. Na *Regressão Quadrática*, a curva para a qual se calcula os valores das constantes é a gerada por uma função quadrática, isto é, uma parábola. Seu objetivo é encontrar as constantes **a**, **b** e **c** da equação de parábola $f(x) = ax^2 + bx + c$.

4.6.4 DINÂMICO

As técnicas de tipo Dinâmico escolhem dinamicamente como técnica de predição aquela, dentre as que para ele estão disponíveis, que ofereça o melhor desempenho. Se uma dessas técnicas possui parâmetros, eles são ajustados, isto é, procuram-se valores de parâmetros que ofereçam o melhor desempenho. Para realizar esse ajuste, utilizam-se a própria série de entrada e, no ajuste de técnicas da família KNN, as demais séries da base de medidas.

Inicialmente, antes de sua execução, a técnica dinâmica precisa conhecer as técnicas que ela irá calibrar. A versão atual do Peixe-Dipnoico conta com as seguintes técnicas já apresentadas nas seções anteriores: *KNN*, *KNN2*, *KNN3*, *Médias Móveis*, *Regressão Linear* e *Regressão Quadrática*. Todas essas estão habilitadas, assim como qualquer outra que venha a ser implementada, a figurar na lista de técnicas. Se utilizadas, as técnicas do tipo *KNN* são definidas inicialmente com $k = 1$, utilizando, por gerarem em experimentos os melhores desempenhos, a distância Euclidiana como métrica de distância e a mediana como medida de tendência central. As técnicas do tipo *Autorregressão* são definidas inicialmente com *window* = 1, também por razão dos resultados em experimentos.

Na fase de calibragem, todas as técnicas candidatas à escolha são executadas, variando seu respectivo parâmetro. Para as técnicas do tipo *KNN*, seu valor de *k* é incrementando iterativamente em uma unidade, enquanto para as técnicas do tipo *Autorregressão*, a variação ocorre no valor de *window*, também incrementado iterativamente em uma unidade. As técnicas do tipo *AjusteDeFunções* não são calibradas, pois não possuem parâmetros. Para

cada tipo de técnica, um processo de calibragem específico é utilizado, já que as heurísticas são diferentes.

Para a calibragem das técnicas do tipo *KNN*, o processo transcorre da seguinte forma:

1. Incrementa o valor de *k* enquanto houver melhora no desempenho.
2. A cada melhora, o valor de *k* que gerou tal resultado é armazenado.
3. Se o valor de *k* não oferecer melhora de desempenho após N execuções, a calibragem é finalizada. O valor de N é definido por: *stopConditionKNNFamily* × | *series* |, onde *stopConditionKNNFamily* é um valor que representa o percentual do número de iterações em relação à quantidade de séries (|*series*|), até o qual o processo de calibragem buscará o valor de *k* que melhore o desempenho. Por exemplo, se a base possui 100 séries e o atributo *stopConditionKNNFamily* for 0,1 (10%), a calibragem é finalizada após 10 execuções que não ofereçam ganho de desempenho.
4. Se o valor de *k* alcançar o número de séries da lista de séries (*series*), a calibragem é finalizada.

Para a calibragem das técnicas do tipo *Autorregressão*, o processo transcorre iterativamente da seguinte forma:

1. Incrementa o valor de *window* enquanto houver melhora.
2. A cada melhora, o valor de *window* que gerou tal resultado é armazenado.
3. Assim que o valor de *window* utilizado não oferecer melhora de desempenho, a calibragem é finalizada.
4. Se o tamanho da janela for maior que o tamanho da série de entrada, a calibragem é finalizada.

Ao final dos dois processos de calibragem, obtêm-se informações sobre o valor do parâmetro (valor de *k* ou de *window*) que gerou o melhor resultado, bem como o valor da métrica de desempenho obtida com o uso desse parâmetro. No caso das técnicas que não possuem parâmetros, é retornada apenas a indicação do desempenho obtido com sua execução. Analisando tais informações e resultados, define-se a técnica a ser utilizada para a predição. Essa técnica é, então, aplicada à entrada e gera os valores preditos, de acordo com o horizonte de predição definido.

O protocolo para calibragem descrito nessa seção é seguido nas variações descritas em suas subseções. Entretanto, a forma como cada série é tratada por essas variações, as técnicas utilizadas e a averiguação do desempenho diferem de uma para a outra. As Subseções 4.6.4.1 e 4.6.4.2 apresentam essas variações.

4.6.4.1 DINÂMICO LOCAL

A técnica *Dinâmico Local* é uma variação da técnica dinâmica. Nele, o desempenho das técnicas é calculado localmente, utilizando valores da própria série de entrada. Para verificar o ganho ou a perda de desempenho, é separado da série de entrada um conjunto de valores efetivos com tamanho igual ao horizonte de predição definido pela técnica dinâmica. O remanescente da série de entrada, após a retirada dos valores efetivos, é utilizado como entrada da técnica em processo de calibragem. Após a execução da técnica com a série de entrada fornecida, o último valor predito é comparado com o último valor efetivo, sendo, em seguida, tomada a decisão de continuar ou não as iterações em busca da melhor configuração para a técnica em processo de ajuste.

Para exemplificar, considere-se o exemplo da Figura 36, supondo um horizonte de predição igual a 1. Considere-se também, para simplificação do exemplo, que somente as técnicas Regressão Linear e Médias Móveis são candidatas a serem escolhidas como preditoras. Na Série 1, a técnica separa o último valor (valor efetivo), o número 5, e utiliza a restante (isto é, a série {7,7,5,5}) no processo de calibragem. É fácil observar, pela visualização na figura, que a técnica Médias Móveis com tamanho de janela igual a 1 tem desempenho ideal (erro nulo) ao realizar a predição com o remanescente da Série 1, pois o valor predito também seria 5. A técnica Regressão Linear teria desempenho inferior, apresentando 4 como valor predito. Já com relação à Série 2, o cenário seria o oposto, sendo a Regressão Linear a técnica que obteria o desempenho ideal para a série usada no processo de calibragem, pois o valor predito seria 18. A Médias Móveis apresentaria o valor 16 como a melhor predição (usando janela igual a 1).

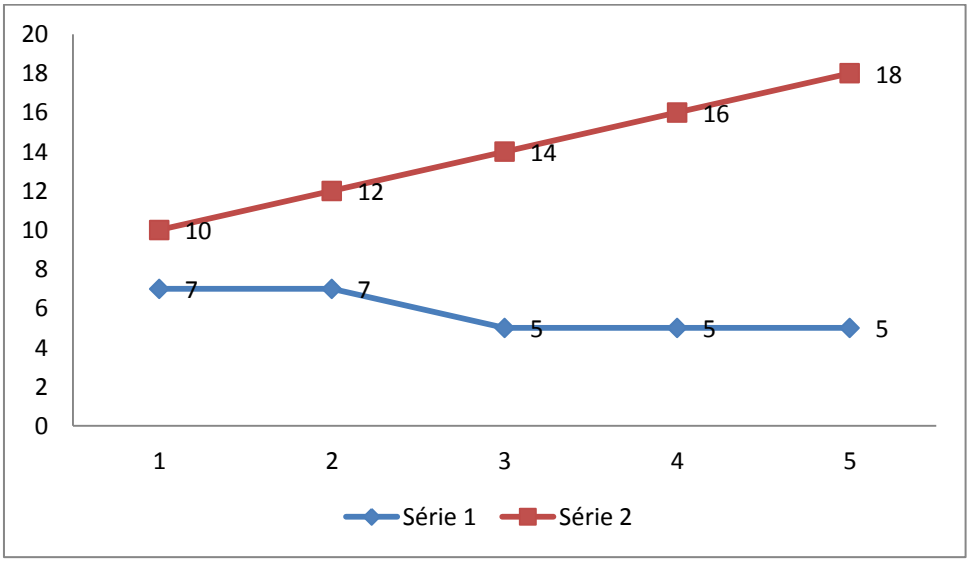

Figura 36. Séries fictícias utilizadas em uma execução da técnica Dinâmico Local.

4.6.4.2 DINÂMICO GLOBAL

A técnica *Dinâmico Global* é também uma variação da técnica dinâmica. Nela, o desempenho das técnicas é calculado globalmente, isto é, avalia-se quão eficaz é a técnica levando-se em consideração seu desempenho médio na predição de todas as séries da lista de séries da técnica dinâmica. Após esse processo, o Erro Relativo Médio obtido é utilizado, em função de seu incremento ou decremento, para auxiliar na tomada de decisão de continuar ou não as iterações em busca da melhor configuração para a técnica em ajuste.

Para exemplificar, considere-se o exemplo da Figura 37, supondo um horizonte de predição igual a 1. Considere-se também, para simplificação do exemplo, que somente as técnicas Regressão Linear e Médias Móveis são candidatas a serem escolhidas como preditoras. Sendo a Série 1 a série de entrada, a técnica separa os últimos valores das demais séries (valores efetivos) e utiliza o remanescente de cada uma delas no processo de calibragem. Utilizando como preditora a técnica Regressão Linear, os valores preditos para os remanescentes das outras séries são 18 (Série 2) e 13 (Série 3), os quais geram erro nulo. Em contraste, a técnica Médias Móveis com janela igual a 1 gera como valores preditos 16 e 12, os quais se afastam dos valores efetivos em 2 e 1 unidades, respectivamente. Assim, a técnica Regressão Linear é escolhida como preditora para a série de entrada, devido ao seu melhor desempenho no caso geral, gerando como predição o valor 12.

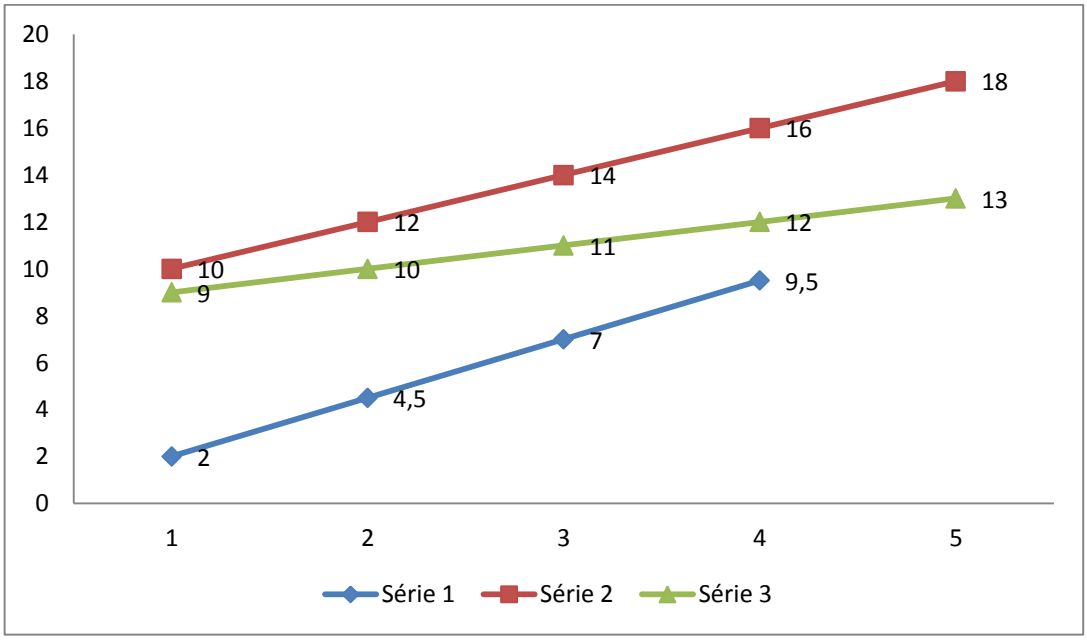

Figura 37. Séries fictícias utilizadas em uma execução da técnica Dinâmico Global.

4.7 APLICAÇÕES PRÁTICAS DO FRAMEWORK

Para demonstrar o uso do framework e suas funcionalidades, uma aplicação foi desenvolvida. Essa aplicação é uma plataforma de experimentos, destinada a prover ao usuário final uma interface, na forma de linha de comando e gráfica, para criação e execução de experimentos. As versões de linha de comando e gráfica da plataforma são representadas, respectivamente, pelas entidades *ExperimentPlatformCLI* e *ExperimentPlatform*.

ExperimentPlatformCLI possui como atributo uma instância da entidade *GerenciadorDeExperimentosPeixeDipnoico*. Essa entidade é uma extensão de *GerenciadorDeExperimentos* e atende a demandas específicas das técnicas de predição até então desenvolvidas, não contempladas pelo processo de execução de experimentos de *GerenciadorDeExperimentos*, mostrado na Seção 4.5. Os métodos implementados nessa extensão oferecem novas funcionalidades aos métodos do ancestral, tendo-os como base. Desse modo, há métodos que, por exemplo, configuram os parâmetros passados para o *Controlador de Saída* de forma a incluir novas perspectivas aos gráficos. Por exemplo, possibilita-se a geração de gráficos contendo as *k* séries mais similares encontradas pela execução das técnicas da família *KNN*, ou gráficos contendo a série gerada pela função de ajuste juntamente com a série de entrada, no caso das técnicas do tipo *Ajuste de Funções*. Além disso, possibilitam a inclusão de implementações alternativas dos experimentos (isto é, da entidade *Experimento*), como as que lidam com as técnicas do tipo *KNN* e do tipo *Dinâmico*.

As implementações alternativas dos experimentos, na versão atual do *framework*, modificam o fluxo de execução de experimentos descrito na Seção 4.2 que lidam com as técnicas da família *KNN* e Dinâmicas. Técnicas como as mencionadas anteriormente utilizam uma lista de séries para sua execução. Portanto, é necessário definir essa lista em algum momento antes do Passo 3 do fluxo de execução de experimentos.

Para avaliar o desempenho de técnicas que se fundamentam em bases de dados (como as do tipo KNN), é necessário usar técnicas de experimentação que reduzam a influência de circunstâncias muito favoráveis ou muito desfavoráveis à obtenção de um bom desempenho. Assim, uma das implementações alternativas dos experimentos foi a que define a metodologia de validação de experimentos *Leave-one-out* (TAN, 2006), a qual, dada uma amostra, separa iterativamente um elemento para o teste do modelo e todos os outros para o treinamento do mesmo. A cada nova iteração, o elemento de teste é transferido ao conjunto de treinamento, e um elemento do conjunto de treinamento torna-se teste, desde que ainda não tenha cumprido

esse papel. Quando no papel de teste, a série será a entrada da técnica e usará as demais (treinamento) para compor a predição. Em cada iteração, os desempenhos das predições são calculados e, ao final do processo, esses valores são sumarizados através, por exemplo, da média aritmética.

Além da técnica *Leave-One-Out*, a versão atual do *framework* implementa uma variação no fluxo de execução do experimento. Essa variação consiste em, após realizar a ordenação da lista de séries utilizada pelo experimento em ordem crescente de tamanho, realizar o experimento utilizando como conjunto de teste uma porcentagem dessa lista correspondente às primeiras posições, isto é, às de menor tamanho. Essa variação assemelha-se à metodologia de validação de experimentos *Holdout* (TAN, 2006), diferenciando-se no fato de realizar a ordenação das séries por tamanho.

A entidade *ExperimentPlatformCLI* pode ser usada na forma de linha de comando, o que é útil para a execução de experimentos em lote. Existem opções para configurar especificamente os parâmetros de cada técnica implementada, definir o projeto de origem, a métrica e o horizonte de predição, gerar gráficos e histogramas, entre muitas outras. Essas e outras opções são listadas a seguir. É importante observar que um valor deve seguir cada uma dessas opções, de modo a definir o atributo representado por ela.

- --repo-path: define o caminho onde o repositório deve ser encontrado.
- --repo-type: define se o repositório analisado é GIT ou SVN.
- --horizon: define o horizonte de predição.
- --metric: define a métrica utilizada para a medição de um projeto ou para a realização de um experimento. Entre os valores disponíveis para essa opção estão *LinesOfCode*, *CyclomaticComplexity*, *ClassInterfaceSize* e *NumberOfAttributes*.
- --algorithm-pred: define a técnica de predição a ser utilizada no experimento. Cada técnica possui seu conjunto de parâmetros associado, listados a seguir.
    - k-NN: --k (número de séries similares), --distMetric (métrica de distância a ser utilizada) e --centTendMeas (medida de tendência central a ser utilizada, isto é, média ou mediana).
    - Médias Móveis: --window (tamanho da janela)
    - Dinâmicos: --stopConditionKNNFamily (porcentagem aplicada sobre o número de séries da base para definir o número de iterações em que a técnica irá tentar obter melhor desempenho)

- --project: define o projeto de onde serão obtidas medidas, tanto para a medição quanto para os experimentos. Se não for definido, serão extraídas as medidas do repositório indicado por --repo-path e o projeto recebe o nome "AnonymousProject".
- --data-inicial: define a data a partir da qual as medidas serão obtidas do repositório. Se não for definida, será considerada a data 1 de janeiro de 1970.
- --data-final: define a data até a qual as medidas serão obtidas do repositório. Se não for definida, será considerada a data atual.
- --min-size: define o tamanho mínimo das séries usadas no experimento. Se não for definida, serão considerados todos os tamanhos de série.
- --output: define o diretório onde os experimentos serão salvos. Se não for definido, será considerado o diretório atual (isto é, aquele de onde se executou a plataforma de experimentos).
- --plot-experiment-graphics: define se serão gerados gráficos para o experimento corrente. Os gráficos contêm a série de entrada, a série efetiva e a série predita.
- --plot-series-graphics: define se serão gerados gráficos contendo as séries da base, todas elas íntegras.
- --plot-histogram: define se será gerado um histograma relacionado ao tamanho das séries.
- --execute: determina a execução dos experimentos com a configuração definida pelas demais opções.
- --extract: extrai medidas de acordo com as definições das opções citadas.

Para servir como *front-end* para *ExperimentPlatformCLI*, existe a entidade *ExperimentPlatform*. Ela dispõe de recursos gráficos para auxílio na inserção de parâmetros para as técnicas e experimentos, exibição das saídas geradas e invocação de funcionalidades disponíveis no framework, como a geração de gráficos e histogramas. A Figura 38 mostra a captura de tela da interface gráfica da plataforma de experimentos.

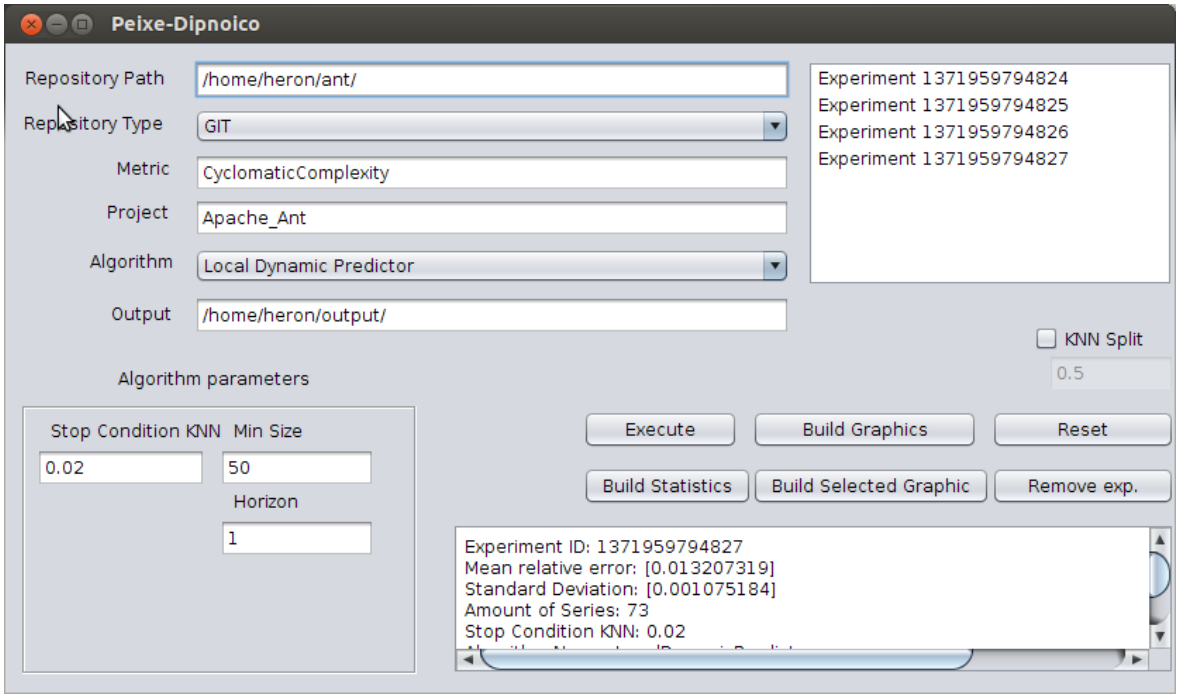

Figura 38. Plataforma de experimentos.

Na figura, os componentes visuais de interação com o usuário são utilizados para levar à aplicação os valores dos parâmetros identificados pelo rótulo localizado sobre eles. No canto superior direito da janela da aplicação, há uma lista onde as instâncias dos experimentos ficam representadas através de seus números de identificação. A área branca localizada na parte inferior da janela é uma área de texto destinada a exibir as saídas das execuções dos experimentos e mensagens de erro. Há seis botões que realizam chamadas a *ExperimentPlatformCLI*. Um deles é o botão *Execute*, que adiciona um experimento utilizando como parâmetros os valores preenchidos na interface e em seguida o executa. Os botões *Build Selected Graphic* e *Build Graphics* coletam os resultados dos experimentos e geram gráficos de séries para o experimento selecionado ou para todos os que estão na lista de executados. Ambos geram também histogramas. O botão *Build Statistics* gera o relatório de desempenho do experimento e o exibe na área de texto. O botão *Remove exp.* remove o experimento selecionado na lista. Por fim, o botão *Reset* remove todos os experimentos e retorna os componentes visuais ao seu estado inicial (isto é, ao estado em que se encontravam no início da execução da aplicação).

Em outras aplicações do Peixe-Dipnoico, valores oriundos de predições geradas pelo *framework* podem ser a entrada de abordagens relacionadas à mensuração da qualidade do software que utilizem valores de métricas estruturais. MACIA *et al.* (2012) propõem uma abordagem de detecção de anomalias que, a partir de condições lógicas baseadas nos valores

das métricas, determina se uma classe possui ou não anomalias. No referido artigo, define-se uma anomalia de código como uma violação na arquitetura estabelecida para o sistema em desenvolvimento. Uma das estratégias de detecção de anomalias dessa abordagem é mostrada na Figura 39, na qual AFTD mede o número de acessos a atributos de outras classes, WMC mede a complexidade estrutural de toda a classe e TCC é uma métrica de coesão (*Tight Class Cohesion*). Os *thresholds* mostrados na figura são ajustados para cada contexto de utilização. Utilizando valores preditos através do Peixe-Dipnoico e aplicando esses valores em uma estratégia de detecção como a que foi mostrada na Figura 39, é possível obter predições da detecção de anomalias para um ou mais instantes futuros.

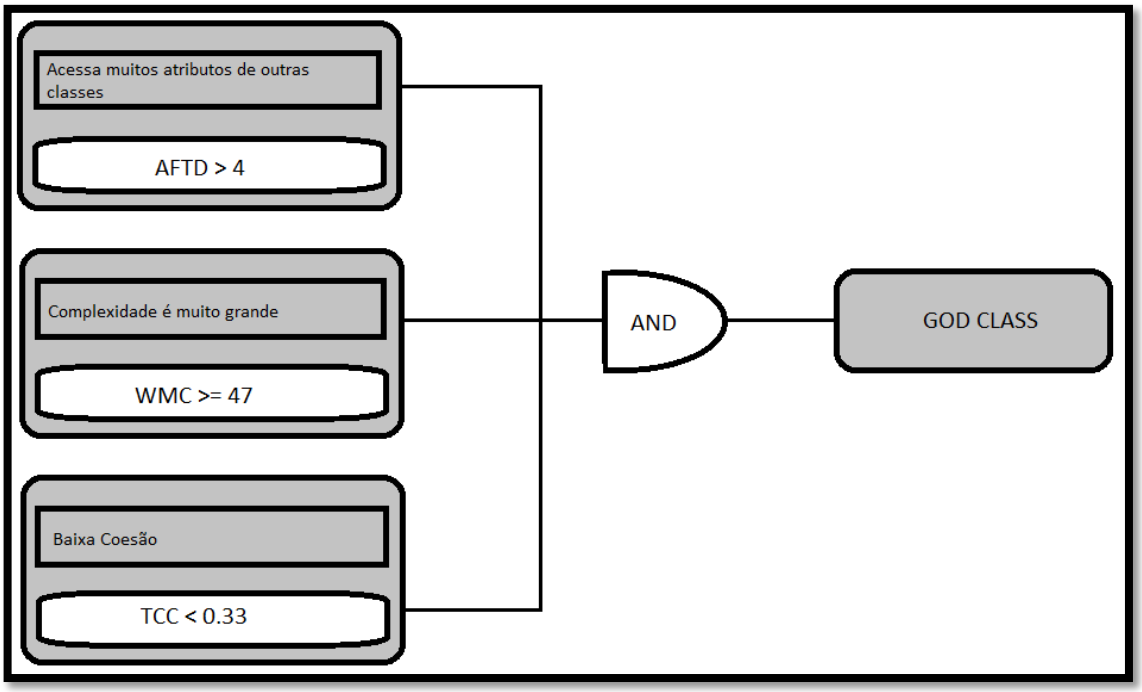

Figura 39. Estratégia de detecção de anomalias, adaptado de MACIA *et al.* (2012).

4.8 CONSIDERAÇÕES FINAIS

O framework proposto objetiva oferecer à área de pesquisa sobre predição com métricas de software uma base para a realização de experimentos, obtenção de medidas e visualização de resultados. Através de pontos de extensão (*hotspots*), como a entidade *Predictable*, a versão atual do framework possibilita a inclusão facilitada de novas técnicas. Outras entidades, embora não sejam pontos de extensão, são suscetíveis a serem especializadas de modo a atender a demandas específicas de uma aplicação, como a demanda por novas formas de realizar experimentos, conforme descrito na Seção 4.7. Dessa forma, o framework se torna útil a variados propósitos, como a predição de medidas (utilizando diretamente as técnicas de predição), gerenciamento de experimentos em predição de medidas

e predição de anomalias (MACIA; ARCOVERDE; *et al.*, 2012), conforme citado na Seção 4.7, dentre outros.

CAPÍTULO 5 – AVALIAÇÃO EXPERIMENTAL

5.1 INTRODUÇÃO

O Capítulo 4 apresentou a proposta de um *framework* que viabilize a implementação de novas técnicas de predição e auxilie na execução de experimentos. Esse *framework* visa prover meios de obtenção de dados para as séries temporais, de gerenciamento dos experimentos e de visualização dos resultados obtidos com as técnicas desenvolvidas.

Neste capítulo, apresenta-se o resultado da utilização do *framework* como suporte para a realização de experimentos com as técnicas apresentadas no Capítulo 4, como descrito na Seção 5.2. Os dados utilizados nesses experimentos são obtidos de projetos *open-source* maduros e de alta popularidade, mostrados na Seção 5.3. A Seção 5.4 apresenta as métricas de avaliação de desempenho, utilizadas para indicar a eficácia das técnicas implementadas, cujos resultados são apresentados e analisados na Seção 5.5. Na Seção 5.6, realiza-se uma discussão sobre as ameaças à validade das conclusões obtidas a partir dos resultados dos experimentos. Por fim, a Seção 5.7 apresenta as considerações finais.

5.2 DESCRIÇÃO DOS EXPERIMENTOS

Os experimentos realizados comparam os desempenhos das técnicas novas, desenvolvidas durante esta pesquisa (KNN e Dinâmico), com as técnicas clássicas de predição (Médias Móveis e Regressão Linear), todas elas descritas no Capítulo 4. Para a execução dos experimentos, foram selecionados projetos *open-source* de alta popularidade. Esses projetos são gerenciados pelo sistema de controle de versões *Git*, o que possibilita maior velocidade na coleta de medidas, por ser realizada em um clone local do repositório. Por restrições da biblioteca *Oceano* (Capítulo 4), os projetos onde residem os artefatos submetidos à medição devem ser gerenciados pelo sistema *Maven* e escritos na linguagem de programação Java.

Antes do início dos experimentos, sucedeu-se uma fase de coleta de medidas, as quais desempenham o papel de **variáveis independentes** das técnicas. Essa coleta foi realizada com o próprio Peixe-Dipnoico, conforme explicado no Capítulo 4. Algumas restrições foram aplicadas ao processo de coleta, de modo que somente código-fonte escrito até a versão 1.6 da linguagem Java fosse considerado. Isso foi necessário devido às limitações de algumas ferramentas de medição, utilizadas pelo Oceano, em reconhecer sintaxes próprias da versão atual (1.7), o que estava gerando medidas incorretas. Por esse motivo, foram consideradas

medidas obtidas de artefatos cujas datas das revisões fossem anteriores ao dia **7 de julho de 2011**, data de lançamento da versão 1.7 da linguagem Java.

As variáveis independentes obtidas da fase de coleta são relativas a diversos tipos de métricas estruturais. Foram escolhidas métricas com presença marcante em pesquisas relacionadas, como Complexidade Ciclomática, Número de Linhas de Código e Número de Métodos Públicos. Todas elas foram definidas no Capítulo 2.

Conforme já explicado no Capítulo 4, em nível de abstração diferente, a execução de um experimento consiste em:

1. Obter um conjunto de séries temporais de um determinado projeto.
2. Executar uma técnica de predição sobre cada série temporal, configurada com um conjunto de parâmetros definido pelo usuário. Cada série é dividida em duas subséries: a parte contendo os valores mais recentes, de tamanho igual ao horizonte de predição, chamada de **série efetiva** e a outra parte, chamada de **série de entrada**. A subsérie usada como entrada para a técnica de predição é a **série de entrada**, enquanto a **série efetiva** é comparada posteriormente à série de valores preditos, para cálculo dos erros de predição, como é mostrado na Seção 5.4.
3. Calcular o erro global da técnica, utilizando os erros de predição calculados no passo 2, como é mostrado na Seção 5.4.

A comparação entre valores efetivos e preditos, descrita no passo 2 do processo de execução de um experimento, acontece entre valores correspondentes de ambas as subséries. Por exemplo, seja a série $S = \{s_1,s_2,s_3,s_4,s_5\}$. Considerando um horizonte de predição igual a 2, seriam geradas as subséries $S_{efetiva} = \{s_4,s_5\}$ e a subsérie de entrada $S_{entrada} = \{s_1,s_2,s_3\}$. Após a execução da técnica de predição utilizando $S_{entrada}$, seria gerada a série de valores preditos $S_{predita} = \{s_4', s_5'\}$. Assim, os erros contabilizados seriam aqueles entre s_4 e s_4', considerado o erro do primeiro valor predito, e entre s_5 e s_5', considerado o erro do segundo valor predito. Os procedimentos para o cálculo desses erros são apresentados na Seção 5.4.

5.3 PROJETOS DE SOFTWARE EM AVALIAÇÃO

Conforme explicado na seção 5.2, os experimentos foram executados com base em projetos *open-source* de alta popularidade, gerenciados pelo sistema de controle de versões *Git* e escritos na linguagem de programação Java. Todos esses projetos estão hospedados no

repositório *Github*[21] e pertencem ao *Apache Software Foundation*. São eles: *Maven 3*[22], *Ant*[23] e *Tomcat*[24]. Nos parágrafos seguintes, são apresentadas algumas informações sobre esses projetos. As informações são obtidas dos sites oficiais desses projetos e do Ohloh[25], um diretório de buscas *online* de informações relacionadas a projetos *open-source*, como popularidade, linguagens utilizadas, estatísticas de atividades (*commits* por tempo e por desenvolvedor) e contagem de linhas de código, linhas em branco e linhas de comentários ao longo do tempo.

O *Maven* é um sistema de gerenciamento de construção (do inglês, *build management system*) baseado no conceito de "modelo de objetos do projeto" (*POM*). Todas as informações relativas à construção, à geração de relatórios e à documentação de um projeto são gerenciadas por ele a partir de um ponto central, um arquivo *XML* que lista dependências, repositórios onde as dependências podem ser encontradas, plug-ins de utilidades para o processo de construção e configurações da construção. Segundo o site Ohloh, o projeto, em sua terceira versão, possui alto nível de atividade. Com mais de 20 mil *commits* realizados por mais de 80 contribuintes, o projeto totaliza aproximadamente 332 mil linhas de código. É predominantemente escrito em Java, estável e mantido por uma equipe de desenvolvimento grande.

O *Ant* é uma biblioteca Java que pode ser usada também como ferramenta de linha de comando. Seu uso principal é também apoiar o gerenciamento de construção de aplicações Java, facilitando sua compilação, montagem, teste e execução através da descrição de processos de construção em um arquivo XML. *Ant* também pode ser utilizada para construir aplicações escritas em outras linguagens, como C/C++. Segundo o site Ohloh, o projeto possui um nível moderado de atividade. Com mais de 12 mil *commits* realizados por quase 50 contribuintes, totaliza aproximadamente 262 mil linhas de código. É predominantemente escrito em Java, estável e mantido por uma equipe de desenvolvimento grande.

O *Tomcat* é uma implementação de código aberto das especificações *Java Servlet* e *JavaServer Pages* (JSP), ambas desenvolvidas sob a especificação do programa de padronização para a tecnologia Java, o *Java Community Process* (JCP). Além de implementar essas especificações, inclui também funcionalidades que ajudam no desenvolvimento e na implantação de aplicações para a internet e *web services*. Segundo o site Ohloh, o projeto

[21] https://github.com/
[22] http://maven.apache.org/
[23] http://ant.apache.org/
[24] http://tomcat.apache.org/
[25] http://www.ohloh.net/

possui um nível muito elevado de atividade. Com mais de 26 mil *commits* realizados por mais de 45 contribuintes, totaliza aproximadamente 4,8 milhões de linhas de código. É predominantemente escrito em HTML (Java corresponde a 15%), estável e mantido por uma equipe de desenvolvimento grande.

5.4 MÉTRICAS DE AVALIAÇÃO DE DESEMPENHO

Toda predição possui associada a si um erro, o qual só pode ser averiguado com o passar do tempo, quando o acontecimento predito se suceder de fato. Por exemplo, a predição da quantidade de chuvas para o dia seguinte pode indicar que na ocasião haverá 10 mm de precipitação. No entanto, somente no dia seguinte é que se poderá averiguar se e o quanto a predição do dia anterior errou. O resultado dessa averiguação indica a qualidade da predição realizada no dia anterior, dependendo se houve erro e, em caso positivo, da magnitude desse erro.

Em predições numéricas, é necessário encontrar formas de mensurar seu erro, para que se conheça sua precisão e a qualidade da técnica utilizada. A forma mais simples de obter essa medida é através do **Erro Absoluto**. O Erro Absoluto é simplesmente a diferença entre as magnitudes do valor predito e do valor efetivo, isto é, valor que foi efetivamente medido na mesma ocasião indicada pela predição. Usando o exemplo da predição da quantidade de chuvas, se o valor predito for 10 mm e no dia seguinte se verificar que a quantidade real foi 8 mm, então o Erro Absoluto foi de 2 mm. É importante observar que o referencial desse tipo de erro é o valor efetivo, isto é, seu objetivo é medir o quanto a predição se afastou do valor real, tanto acima quanto abaixo. Portanto, a fórmula do **Erro Absoluto** (**EA**) usada nessa dissertação é dada por **(**9**)**, onde x representa o valor efetivo e x’ representa o valor predito.

$$EA = | x - x' | \quad \textbf{(9)}$$

No contexto da predição de medidas estruturais, as disparidades nas magnitudes entre valores efetivos, contidos na série temporal, e valores preditos tende a ser grande. Para valores de uma mesma métrica, obtidos de um determinado projeto ao longo das revisões de suas classes, é possível obter séries com medidas de ordens de grandeza diferentes. Por exemplo, pode-se obter uma série S_1 = {1,2,3,4,5} juntamente com uma série S_2 = {4996,4997,4998,4999,5000}, ambas constituídas de valores referentes à métrica Número de Linhas de Código. Nesse caso, utilizar o Erro Absoluto pode levar a conclusões equivocadas sobre a qualidade das predições. Supondo que predições realizadas para o próximo valor das séries S_1 e S_2 foram, respectivamente, 6 e 5001, e os valores efetivos foram, respectivamente, 7 e 5002, em ambos os casos o valor de EA foi 1, entretanto não é sensato dizer que as duas

estimativas têm a mesma qualidade. A predição realizada para a primeira série teve um erro de uma unidade em um contexto no qual as medidas variam em uma unidade, enquanto a que foi realizada para a segunda série também teve um erro de uma unidade em um contexto no qual as medidas variam em mil unidades. O impacto no primeiro caso é maior. Para tornar a avaliação do erro insensível a essas disparidades, existe o **Erro Relativo** ou **Erro Percentual**. O Erro Relativo (ER) mede o percentual de afastamento do valor predito em relação ao valor efetivo. O ER é dado por **(10)**, onde x representa o valor efetivo e x' representa o valor predito. Observe-se que o numerador de **(10)** é o Erro Absoluto **(9)**. Usando o exemplo desse parágrafo, o ER de S_1 foi $|7-6|/7$ = 0,1428 ou 14,28%, enquanto o ER de S_2 foi $|5002-5001|/5002$ = 0,0002 ou 0,02%.

$$ER = \frac{|x - x'|}{x} \qquad \textbf{(10)}$$

Em (10), o denominador é x, que é o valor efetivo. Quando x tende a zero, o valor de ER tende a infinito. No entanto, é comum ter valores efetivos próximos ou iguais a zero. Nesses casos, mesmo que a predição seja precisa, o ER seria muito alto. Para contornar esse problema, inerente à construção da fórmula, utilizou-se uma variação de (10), baseada em ÜBERHUBER (1997). Nessa variação, quando o valor efetivo é menor que 1 (um), considera-se como ER somente o numerador de (10), isto é, | x – x' |.

Essas duas medidas (Erro Absoluto e Erro Relativo) são adequadas à mensuração do desempenho de uma predição específica. Nessa dissertação, contudo, há o interesse de avaliar o desempenho de uma técnica aplicada a predizer valores de determinada métrica, extraídos dos artefatos de determinado projeto. Nesse caso, a mensuração deve ter uma abrangência de caráter global. Para atender a essa demanda, utilizou-se, nessa dissertação, a média dos Erros Relativos, ou **Erro Relativo Médio (ERM)**. Para calculá-lo, são obtidos os erros relativos das predições que são realizadas em todas as séries utilizadas em um experimento, como descrito na seção 5.2. Em seguida, obtém-se a média aritmética desses erros relativos, como mostra **(11)**, onde ER_i é o erro relativo da predição em uma das N séries utilizadas em um experimento.

$$ERM = \frac{\sum_{i=1}^{N} ER_i}{N} \qquad \textbf{(11)}$$

Quanto menor o valor de ERM, mais eficiente é a técnica avaliada, visto que em média a qualidade das predições em relação ao erro relativo é alta, ou seja, o erro relativo de

cada predição é, em média, baixo. Como acontece com toda média, o valor de ERM não reflete totalmente as características da amostra de resultados de predições, visto que amostras com médias iguais podem ter seus valores em diferentes graus de dispersão em relação à média. Para medir esse grau de dispersão, calcula-se o **Desvio Padrão (DP)**, definido como em **(12)**, onde ERM é o erro relativo médio, x_i são os erros relativos da amostra de resultados e N é o número de resultados da amostra.

$$DP = \sqrt[2]{\frac{(x_1 - ERM)^2 + (x_2 - ERM)^2 + ... + (x_N - ERM)^2}{N - 1}} \quad \textbf{(12)}$$

5.5 ANÁLISE DOS RESULTADOS DOS EXPERIMENTOS

Nesta seção, são apresentadas tabelas com os desempenhos das técnicas implementadas e avaliadas através do *framework*. Nessas tabelas, os valores apresentados correspondem aos valores da métrica Erro Relativo Médio (ERM)[26], descrita na Seção 5.4, obtidos dos resultados dos experimentos realizados para um determinado projeto, utilizando valores de determinada métrica, para cada horizonte de predição, o qual varia entre 1 e 10. As tabelas possuem a seguinte estrutura:

- Técnica: coluna que lista as dez técnicas submetidas a experimentos. Nas tabelas, não serão mostradas as parametrizações que geraram os ERMs relativos a cada uma dessas técnicas, contudo elas estarão presentes no Apêndice C. Todas as técnicas apresentadas nessa coluna foram apresentadas no Capítulo 4.
- H_1 – H_{10}: colunas que representam o ERM dos experimentos realizados, em valores percentuais, para cada horizonte de predição H_i. Os valores contidos nas células correspondentes a essas colunas é o menor ERM gerado para a técnica referente, considerando o horizonte *i*. Por exemplo, o valor da célula cuja linha é a técnica KNN-IR (apresentada no Capítulo 4) e a coluna é H_2 é o menor ERM encontrado nos experimentos realizados para a avaliação do desempenho das predições da técnica KNN-IR quando o horizonte de predição é dois.
- Melhor: Contagem do número de vezes em que cada técnica obteve o melhor desempenho entre todas as outras na tabela, considerando cada coluna. Por exemplo, se uma técnica tem o valor de MELHOR igual a 5, ela teve o melhor

[26] Os Desvios Padrões relacionados a cada ERM são apresentados no Apêndice A.

desempenho nos experimentos para 5 dos 10 horizontes considerados nesta pesquisa.

Na apresentação das tabelas, as técnicas estão classificadas em ordem decrescente com relação ao valor da coluna MELHOR. Em caso de empate, aparece em primeiro lugar aquela que, dentre as empatadas, vence no horizonte mais distante. Se mesmo assim houver empate, considera-se campeã aquela que vence no horizonte imediatamente anterior ao mais distante, repetindo-se o processo até que se encontre um vencedor. Persistindo o empate, ele é assumido e se ordena o nome das técnicas em ordem crescente. Dessa forma, as primeiras técnicas são aquelas que possuem maior poder preditivo no contexto da métrica e do projeto em que foram aplicadas. Além disso, em cada coluna, o menor valor de ERM é destacado com negrito (em caso de empate, todos os valores iguais são destacados).

Observando os resultados apresentados nas tabelas desta seção, pode-se notar a relevância de algumas técnicas sobre outras. Nas 8 comparações, as técnicas Regressão Linear, Regressão Quadrática, Dinâmico Local e KNN não só figuraram sempre entre as últimas posições quanto ao ERM em todos os horizontes como também apresentaram, em geral, valores de ERM muito maiores que os demais. O valor de MELHOR em todas as tabelas para essas técnicas é zero. Da mesma forma, algumas técnicas sempre figuraram nas primeiras posições quanto ao ERM em todos os horizontes. Não houve uma técnica que apresentasse o melhor desempenho em todas as tabelas e em todos os horizontes, no entanto as técnicas KNN-IR, KNN-IA e Médias Móveis se revezam entre as técnicas com melhor desempenho ao longo dos horizontes, como se percebe pelo valor de MELHOR.

Tabela 4 – Projeto: Apache Ant / Métrica: Número de Métodos Públicos.

TÉCNICA	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8	H_9	H_{10}	MELHOR
KNN-IR	**1,05**	**1,90**	**4,36**	5,41	**6,23**	**7,87**	**8,84**	**9,48**	23,00	23,64	7
KNN-IA	**1,05**	**1,90**	**4,36**	**5,38**	6,25	7,96	8,89	9,57	**22,99**	23,69	5
Méd. Móveis	**1,05**	**1,90**	4,39	5,42	6,25	7,96	8,89	9,58	23,07	**23,50**	3
Local KNN	**1,05**	**1,90**	**4,36**	5,42	6,25	7,96	8,89	9,58	23,05	23,98	3
Global	**1,05**	**1,90**	4,39	5,43	6,25	8,01	8,93	9,73	23,14	24,10	2
Global KNN	**1,05**	**1,90**	4,39	5,42	6,31	7,96	8,89	9,98	23,15	24,39	2
KNN	11,37	12,23	16,89	18,47	20,08	23,71	24,30	27,28	29,86	30,08	0
Local	1,51	2,70	4,97	6,43	14,00	16,97	19,72	11,52	30,15	26,85	0
Reg. Lin.	12,39	13,93	15,73	17,72	20,23	23,31	26,92	31,46	37,53	38,72	0
Reg. Quad.	13,67	17,75	22,55	28,87	31,22	34,70	34,90	29,03	54,49	59,92	0

Os resultados da Tabela 4 e da Tabela 5, referentes à métrica Número de Métodos Públicos para os projetos Ant e Tomcat, respectivamente, apresentam um padrão similar em termos dos valores de MELHOR e das técnicas campeãs. Em ambos os casos, KNN-IR e KNN-IA disputam entre si as primeiras posições, embora se perceba que KNN-IR predomina

ao longo dos horizontes. Na coluna H_{10} da Tabela 4, Médias Móveis vence, mas não se pode dizer que as variantes de KNN não realizam boas predições quando o horizonte é mais longo. A análise dos arquivos de resultados dos experimentos referentes a essa coluna revela que, de todas as 143 predições realizadas, Médias Móveis foi melhor que KNN-IR em apenas 8 casos. A prevalência de Médias Móveis sobre KNN-IR deveu-se a um caso em que os valores da série efetiva passam de 16 para 0 (zero). A técnica Médias Móveis, com janela igual a 8, gera uma predição de valor menor do que o último de uma determinada série (Figura 40), como é natural a essa técnica ao realizar predições de séries crescentes como essa. Dessa forma, a técnica ficou mais próxima de zero do que KNN-IR, que seguiu uma tendência constante, como determinado pela análise das séries similares escolhidas por esta última. A diferença de ERM entre essas duas técnicas (Figura 40) foi de 2,61, isto é, 261% (ER de 1600% do KNN-IR subtraído de ER de 1339% de Médias Móveis), o que prejudicou o desempenho de KNN-IR em relação a Médias Móveis. Se essa série S não estivesse presente, mesmo com os 7 casos favoráveis à Médias Móveis, o ERM do KNN-IR seria menor em uma magnitude de aproximadamente 0,02, ou 2%. Situações similares a essa podem ser verificadas na coluna H_{10} da Tabela 11, na coluna H_6 da Tabela 5, na coluna H_{10} da Tabela 8 e nas coluna H_6 e H_8 da Tabela 9, onde determinada técnica aparece isoladamente como a campeã devido ao melhor desempenho na predição de algumas poucas séries.

Tabela 5 – Projeto: Apache Tomcat / Métrica: Número de Métodos Públicos.

TÉCNICA	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8	H_9	H_{10}	MELHOR
KNN-IR	**1,56**	**2,20**	3,42	**3,95**	5,51	6,16	**6,08**	**7,24**	**8,05**	**10,21**	7
KNN-IA	**1,56**	**2,20**	**3,41**	**3,95**	**5,49**	6,16	6,11	7,26	8,11	10,25	5
Méd. Móveis	**1,56**	**2,20**	3,42	3,97	5,51	**6,09**	6,11	7,26	8,11	10,25	3
Local KNN	**1,56**	**2,20**	3,42	3,97	**5,49**	6,16	6,11	7,26	8,11	10,25	3
Global KNN	**1,56**	2,28	3,44	3,97	**5,49**	6,24	6,11	7,26	8,11	10,31	2
Global	**1,56**	2,28	3,44	4,38	5,82	6,17	6,62	7,64	8,45	10,35	1
KNN	15,97	16,46	18,60	19,89	21,02	21,69	22,84	24,35	25,72	26,08	0
Local	2,27	2,78	6,00	6,03	9,52	10,29	12,38	12,68	14,24	19,59	0
Reg. Lin.	30,32	32,37	34,51	36,58	38,67	40,64	42,50	44,44	46,32	48,32	0
Reg. Quad.	35,60	41,46	48,01	55,02	62,79	70,86	79,82	88,95	98,11	107,8	0

Tabela 6 – Projeto: Apache Ant / Métrica: Número de Linhas de Código.

TÉCNICA	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8	H_9	H_{10}	MELHOR
KNN-IA	**1,49**	**2,89**	**5,59**	**6,85**	**8,23**	10,91	13,21	14,43	15,29	16,39	5
KNN-IR	1,53	2,94	5,64	6,88	8,28	**10,88**	13,08	**14,26**	**15,22**	**16,36**	4
Global	1,52	2,90	5,65	7,07	8,31	11,36	**13,07**	14,76	15,81	16,86	1
Global KNN	1,52	2,90	5,65	7,02	8,31	11,04	13, 27	14,44	15,57	16,44	0
KNN	10,59	12,16	13,98	15,23	16,76	17,70	18,82	20,18	20,74	22,54	0
Local	3,16	4,68	7,90	9,87	11,24	12,95	17,13	16,87	19,99	20,60	0
Local KNN	1,51	2,90	5,65	6,87	8,31	10,89	13,25	14,45	15,30	16,38	0
Méd. Móveis	1,55	3,00	5,76	7,07	8,50	11,13	13,42	14,65	15,57	16,69	0
Reg. Lin.	11,59	12,94	14,42	15,84	17,55	19,46	21,38	23,27	25,20	27,15	0
Reg. Quad.	12,67	15,70	19,27	22,67	26,29	31,17	36,31	41,82	49,61	57,98	0

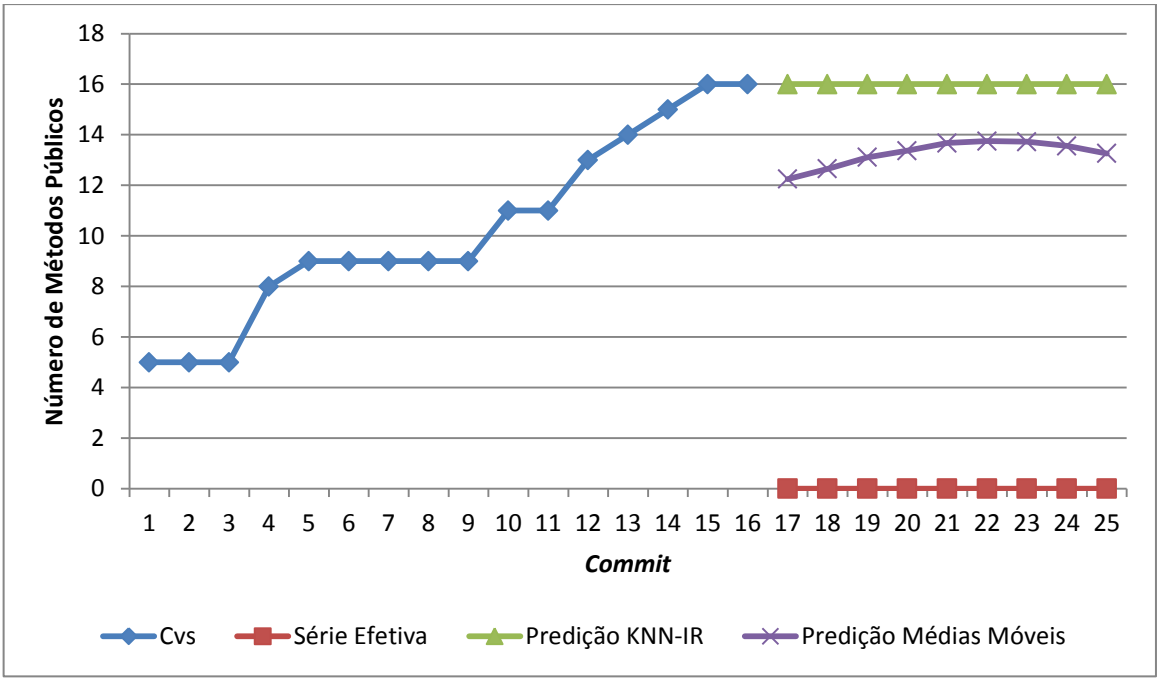

Figura 40 – Comparação da predição realizada por Médias Móveis e KNN-IR para a classe Cvs.

Tabela 7 – Projeto: Apache Maven-3 / Métrica: Número de Linhas de Código.

TÉCNICA	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8	H_9	H_{10}	MELHOR
Méd. Móveis	17,06	21,03	24,92	**29,28**	37,39	**42,40**	**42,31**	45,68	57,60	**63,76**	4
Global	17,06	21,03	24,92	**29,28**	37,61	**42,40**	**42,31**	48,13	60,04	64,21	3
KNN-IA	**16,94**	21,14	**24,78**	29,31	**37,23**	43,96	43,28	45,78	57,13	65,12	3
KNN-IR	17,03	**20,93**	24,79	**29,28**	37,54	43,49	42,87	45,68	57,41	64,36	2
Global KNN	17,19	21,45	25,49	30,09	38,75	45,59	44,74	46,71	**57,08**	66,55	1
Local KNN	17,72	21,61	25,50	29,96	38,72	45,32	43,07	**45,60**	57,59	65,40	1
KNN	50,90	53,48	56,15	57,64	60,93	63,68	68,07	71,81	77,45	76,39	0
Local	19,46	23,81	27,49	31,69	45,68	49,62	46,87	50,24	58,59	70,52	0
Reg. Lin.	55,53	58,72	62,01	65,46	68,66	70,91	73,67	77,13	79,68	82,08	0
Reg. Quad.	43,22	47,37	51,18	53,98	60,66	64,83	65,92	71,03	78,83	87,76	0

Tabela 8 – Projeto: Apache Tomcat / Métrica: Número de Linhas de Código.

TÉCNICA	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8	H_9	H_{10}	MELHOR
KNN-IA	**1,08**	**2,17**	3,28	**4,30**	**5,46**	**6,44**	**7,02**	**7,82**	**8,94**	9,46	8
Local KNN	1,12	2,23	3,39	4,35	5,56	6,48	7,15	7,95	8,96	**9,16**	1
KNN-IR	1,10	2,19	**3,26**	4,32	5,54	6,49	7,14	7,92	9,11	9,72	1
Méd. Móveis	**1,08**	2,19	3,33	4,39	5,72	6,58	7,21	8,02	9,20	9,85	1
Global	1,14	2,19	3,32	4,36	5,55	6,49	7,14	7,91	9,03	9,47	0
Global KNN	1,11	2,23	3,35	4,36	5,55	6,49	7,14	7,90	9,03	9,47	0
KNN	9,88	10,29	10,98	11,90	12,31	13,07	13,71	14,36	14,99	16,22	0
Local	1,37	3,38	4,85	5,58	6,36	8,22	9,33	9,31	11,51	11,66	0
Reg. Lin.	7,27	7,93	8,60	9,31	10,08	10,82	11,65	12,47	13,37	14,14	0
Reg. Quad.	7,47	8,99	10,58	12,32	14,33	16,25	18,53	20,70	23,85	25,74	0

Os resultados da Tabela 6 e da Tabela 8, referentes à métrica Número de Linhas de Código, assim como os experimentos com os valores da métrica Número de Métodos Públicos para os projetos Ant (Tabela 4) e Tomcat (Tabela 5), respectivamente, seguem um padrão de ter KNN-IR e KNN-IA como melhores técnicas em praticamente todos os horizontes. Fogem a esse padrão a coluna H_7 da Tabela 6 e a coluna H_{10} da Tabela 8. No primeiro caso, a técnica Dinâmico Global apresentou os melhores resultados. Observando os resultados dos experimentos realizados para o horizonte 7, KNN-IR, com K igual a 5 e com K igual a 8 (ambos utilizando Distância Euclidiana e Mediana), apresentou os melhores resultados. Os resultados do experimento para o horizonte 7 com a técnica Dinâmico Global mostra que apenas foram escolhidas essas configurações do KNN-IR como preditoras. Devido a isso, a técnica Dinâmico Global teve melhor resultado que KNN-IR, pois escolheu configurações do KNN-IR que tiveram os melhores desempenhos em seus respectivos experimentos. Em outros casos, percebe-se que Dinâmico Global escolhe configurações das técnicas que não são campeãs, o que aumenta a chance de que seu desempenho seja prejudicado. No segundo caso, o desvio ocorre pelo mesmo motivo daqueles ocorridos nos casos citados no parágrafo anterior, ou seja, devido ao melhor desempenho na predição de algumas poucas séries.

Tabela 9 – Projeto: Apache Ant / Métrica: Complexidade Ciclomática.

TÉCNICA	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8	H_9	H_{10}	MELHOR
KNN-IR	**1,98**	**4,11**	**6,62**	7,85	**9,01**	10,46	**12,50**	12,92	**13,05**	14,13	6
Global	1,99	**4,11**	**6,62**	7,89	**9,01**	**10,45**	12,52	13,06	13,10	14,21	4
KNN-IA	2,03	4,14	6,67	**7,83**	9,03	10,50	**12,50**	12,92	13,07	**14,11**	3
Méd. Móveis	2,03	4,15	6,67	7,89	9,06	10,52	12,53	**12,85**	13,07	14,13	1
Global KNN	2,03	4,15	6,67	8,00	9,06	10,51	12,53	12,94	13,08	14,16	0
KNN	9,03	11,09	11,93	13,16	13,46	14,75	15,85	16,96	17,73	18,27	0
Local KNN	2,03	4,15	6,67	7,88	9,06	10,52	12,53	12,94	13,07	14,13	0
Local	3,32	5,82	8,51	9,34	10,91	12,16	13,89	17,39	18,82	20,28	0
Reg. Lin.	14,05	15,62	17,11	18,55	20,00	21,50	22,95	24,34	25,86	27,74	0
Reg. Quad.	12,63	15,78	19,17	22,56	26,13	30,48	35,02	39,13	43,69	50,75	0

Tabela 10 – Projeto: Apache Maven-3 / Métrica: Complexidade Ciclomática.

TÉCNICA	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8	H_9	H_{10}	MELHOR
Méd. Móveis	9,45	10,85	17,85	20,92	27,48	**29,13**	**29,78**	**30,94**	**31,06**	**30,25**	5
KNN-IA	**9,44**	10,81	17,81	**20,88**	**27,46**	29,74	29,91	31,75	32,68	30,95	3
KNN-IR	**9,44**	**10,74**	17,84	20,89	27,48	29,77	29,90	31,73	32,90	30,91	2
Global KNN	9,66	11,30	17,97	20,89	**27,46**	32,02	30,06	31,72	33,50	32,03	1
Local KNN	9,53	10,88	**17,80**	21,01	27,47	29,85	30,12	32,22	32,82	30,56	1
Global	9,45	10,85	17,92	21,02	27,48	31,44	31,93	32,76	31,35	30,75	0
KNN	26,56	28,92	30,12	31,03	32,29	32,95	33,14	33,62	33,25	33,18	0
Local	10,30	16,09	20,82	25,46	30,65	33,02	37,28	37,34	35,28	35,92	0
Reg. Lin.	30,97	33,36	36,12	38,36	40,47	41,73	42,84	43,89	44,12	45,24	0
Reg. Quad.	25,90	31,08	38,32	44,74	51,15	55,76	61,16	65,37	65,19	64,74	0

Tabela 11 – Projeto: Apache Tomcat / Métrica: Complexidade Ciclomática.

TÉCNICA	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8	H_9	H_{10}	MELHOR
KNN-IA	**0,85**	**1,65**	**2,51**	3,46	**4,33**	**4,73**	5,41	6,15	**6,64**	7,25	6
Local KNN	0,87	1,67	2,52	**3,45**	4,36	4,77	**5,40**	**6,13**	6,72	7,29	4
KNN-IR	**0,85**	1,66	**2,51**	3,47	4,37	4,77	5,41	6,15	6,66	7,37	2
Méd. Móveis	0,86	1,67	2,52	3,50	4,37	4,79	5,43	6,18	6,68	**7,18**	1
Global KNN	0,86	1,66	2,55	3,53	4,43	4,84	5,60	6,22	6,74	7,37	0
Global	0,86	1,68	2,56	3,53	4,43	4,84	5,55	6,22	6,74	7,21	0
KNN	7,24	7,76	8,54	9,02	9,18	9,46	10,06	10,37	10,91	11,06	0
Local	1,53	2,47	3,31	4,06	5,04	5,87	6,72	7,96	8,60	8,65	0
Reg. Lin.	7,35	8,00	8,70	9,39	10,04	10,74	11,55	12,26	13,00	13,68	0
Reg. Quad.	5,53	6,71	8,03	9,30	10,72	11,98	13,80	15,78	17,60	20,41	0

Os valores de ERM para a Tabela 7 e para a Tabela 10 se mostraram muito elevados. Isso não significa, no entanto, que as técnicas tiveram desempenho ruim. Em algumas poucas séries, houve em algum momento de sua história uma queda brusca no valor da métrica. A Figura 41 ilustra essas quedas na história de algumas classes como *AssemblyMojo*, *DefaultPluginManager*, *WarMojo*, *PomClassicTransformer* e *ModelUtils*, todas elas representadas por séries temporais. Devido a essas variações bruscas, o desempenho de todas as técnicas fica prejudicado. As técnicas do tipo KNN têm potencial para modelar esse comportamento, no entanto, para que o faça de modo eficiente, é preciso que essa situação de queda seja frequente no projeto. Entretanto, ela acontece em alguns poucos casos, não havendo casos similares que gerassem conhecimento para que as técnicas da família KNN realizassem uma boa predição (pois os valores efetivos não são usados na predição).

Na Tabela 7 e na Tabela 10, Médias Móveis predomina como técnica campeã a partir do horizonte 6. Analisando a evolução dos valores das medidas extraídas do projeto Maven-3, percebe-se que aconteceram alguns casos de quedas bruscas nos últimos valores de algumas séries, como já mostrado pela Figura 41. Por ser essa queda nos últimos valores, mesmo as técnicas mais avançadas, como as da família KNN, não tiveram a oportunidade de modelar esse comportamento. Dessa forma, as técnicas realizaram as predições com os dados em momentos de estabilidade. A Médias Móveis, com sua tendência estacionária, acabou mantendo aproximadamente o último valor antes da queda, ficando sua predição mais próxima do valor efetivo do que a família KNN. Nos casos analisados, a família KNN seguiu sua tendência de crescimento, como determinado a partir da análise das séries similares. No entanto, é importante observar que nenhuma das técnicas teve um bom desempenho nas predições nesses dois casos, devido a essas quedas bruscas.

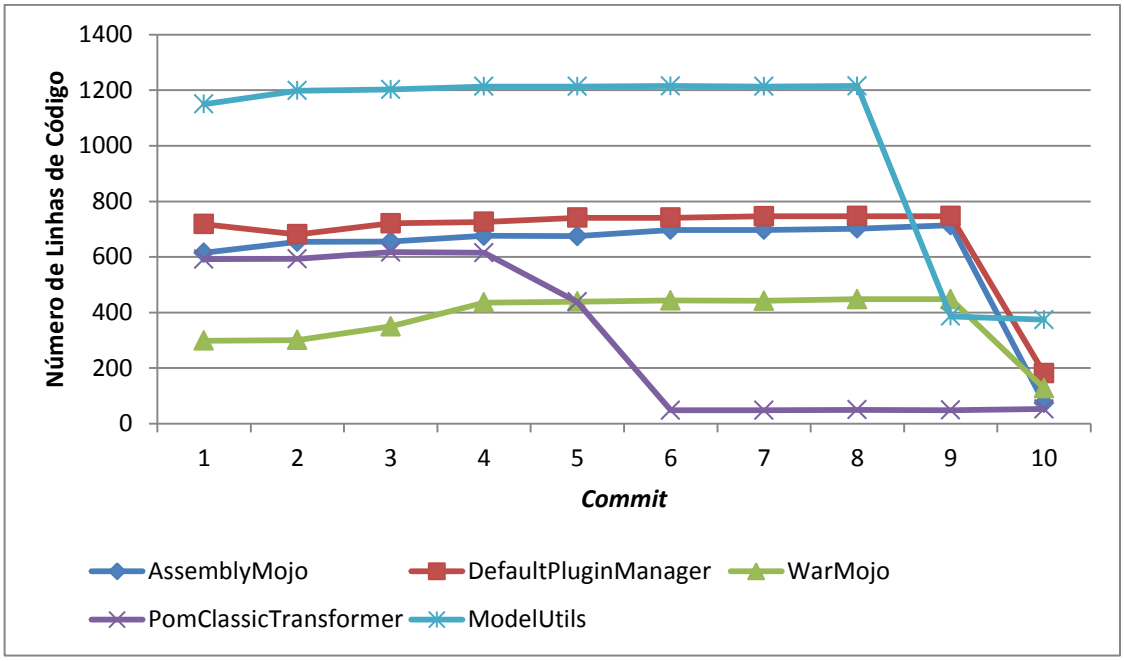

Figura 41 – Quedas bruscas na métrica Número de Linhas de Código, projeto Maven-3.

Considerando o comportamento esperado das predições da técnica Médias Móveis, presume-se que ela tem os melhores desempenhos quando as séries evoluem de modo constante e estacionário, isto é, quando o gráfico dessas séries tem aspecto igual ou próximo de uma reta horizontal (Figura 42). Pela sua concepção de ser a média das complexidades de cada classe, as séries com valores da métrica Complexidade Ciclomática são as que mais predominantemente evoluem dessa forma. Os dados da Tabela 9, Tabela 10 e Tabela 11, referentes a experimentos realizados com séries compostas por valores dessa métrica, reforçam essa hipótese, visto que Médias Móveis aparece no máximo em terceiro lugar no *ranking* de desempenhos de cada coluna H_i. Em contraste às séries formadas por valores da métrica Complexidade Ciclomática, as que são formadas por valores das métricas Número de Linhas de Código e Número de Métodos Públicos evoluem de modo crescente. No primeiro caso, esse crescimento tem, em geral, aspectos de linearidade, isto é, as séries tendem a assemelharem-se a retas (Figura 42). Mesmo em situações de instabilidade, nas quais os valores sofrem um decréscimo brusco, a série volta a assumir o comportamento crescente. No segundo caso, a evolução dos valores alterna estágios de estacionariedade e de crescimento (Figura 42). Nos casos em que as séries são formadas por valores das métricas Número de Linhas de Código e Número de Métodos Públicos, observa-se que Médias Móveis também tem desempenho bom, semelhantemente figurando no máximo na terceira posição do *ranking* de desempenhos, mesmo no horizonte mais longo (Horizonte 10). Embora esse resultado pareça um contrassenso, deve-se considerar que os projetos de onde as medidas foram

extraídas são projetos maduros possuem mais de 10 anos de duração. Por esse motivo, os últimos *commits* realizados são, em geral, de caráter corretivo. Assim, conforme observado, as alterações nos valores das métricas fazem com que eles tenham uma variação pequena, gerando uma evolução com menos variações na história recente das séries. Dessa forma, Médias Móveis também obtém bom desempenho mesmo com métricas cujos valores evoluem de modo crescente.

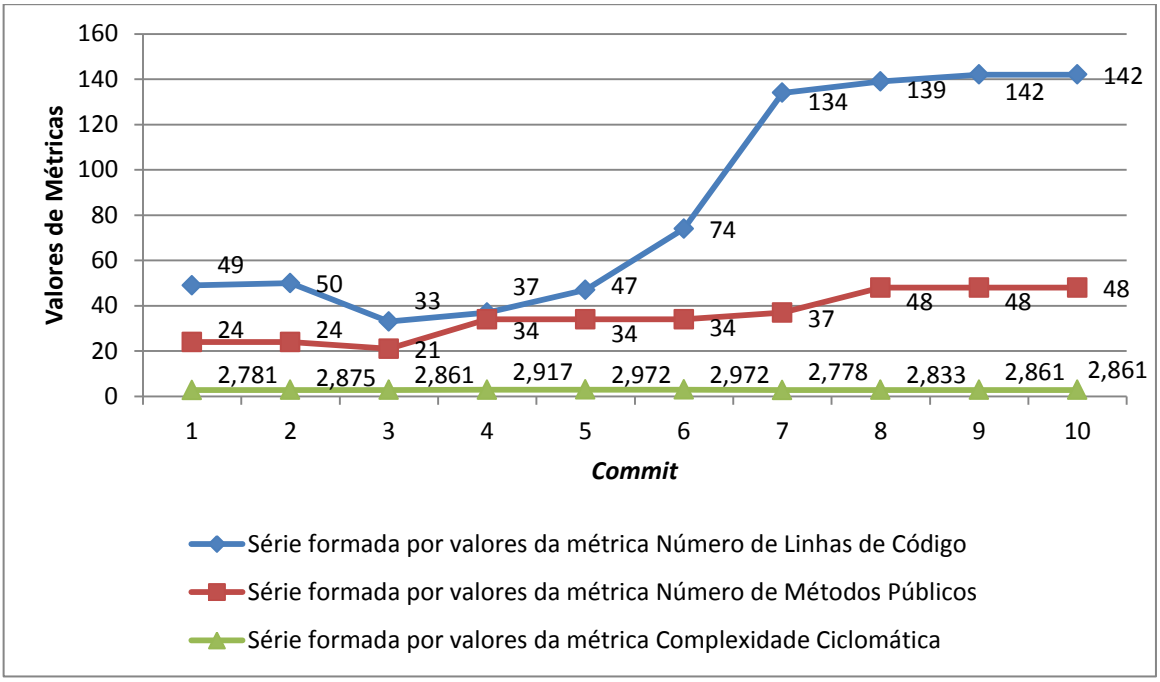

Figura 42 – Comportamento de séries fictícias de acordo com a métrica de que é composta.

5.6 AMEAÇAS À VALIDADE

Segundo WHOLIN et al. (2000), ameaças à validade são questões que podem impactar ou limitar a validade dos resultados de um estudo. Esta seção apresenta as ameaças identificadas na realização dos experimentos, categorizando-as em ameaças à validade interna, ameaças à validade externa, ameaças à validade de conclusão e ameaças à validade de construção, de acordo com definições desse mesmo autor.

As ameaças à validade interna são influências que podem prejudicar o conhecimento sobre o relacionamento de causa e efeito entre o tratamento (as técnicas de predição) e o resultado. Por se tratarem de eventos não controlados pelo pesquisador, podem produzir distorções no resultado esperado. Uma das ameaças à validade interna reside no fato de os dados serem os dados das séries temporais extraídos de projetos de grande porte e externos ao ambiente de trabalho do pesquisador deste trabalho. Dessa forma, não foi possível estudar as características ambientais que causaram variações nos valores. A falta desse conhecimento

dificultou a realização das análises apresentadas na Seção 5.5, principalmente para os casos em que havia um desvio nos padrões de desempenho de determinado conjunto de experimentos. Outra ameaça interna é a granularidade das medidas obtidas, relacionada aos *commits*, o que permitiu que os dados das séries representassem diferentes estilos de desenvolvimento. Por exemplo, o *commit* de um desenvolvedor que o realiza após todo o trabalho do dia tem impacto diferente daquele de um desenvolvedor que o realiza a cada pequena modificação concluída. Dessa forma, duas séries de tamanho igual podem representar evoluções diferentes na geração de valor agregado. Tanto a abordagem quanto os experimentos por ela realizados não levam em consideração esse fato.

As ameaças à validade externa limitam a generalização dos resultados do estudo para outros contextos fora do ambiente avaliado. Neste estudo, esse tipo de ameaça encontra-se em limitações tecnológicas da abordagem, como a limitação à linguagem Java e à utilização de um sistema de controle de versões pelo projeto analisado. Além dessas, também é ameaça à validade externa a quantidade de métricas e a quantidade de projetos analisados, de modo que não se podem generalizar conclusões para outras situações com base apenas nos resultados proporcionado por três métricas extraídas de três projetos. No entanto, é importante observar que esta última ameaça é atenuada pelo fato de esse *framework* ter como um de seus objetivos a viabilização de experimentos relacionados à predição de medidas de qualquer software que atenda aos requisitos da abordagem.

As ameaças à validade de conclusão referem-se a fatos que prejudicam o estabelecimento de relacionamentos estatísticos entre o tratamento e o resultado. Nesta categoria, uma ameaça reside no fato de os valores de ERM gerados pelos experimentos apresentarem diferenças muito pequenas entre si, não possuindo grande relevância estatística para fundamentar conclusões sobre a técnica mais eficiente ou sobre a métrica para a qual as técnicas apresentam melhor desempenho. Ordenando os valores de ERM dos resultados dos experimentos, percebe-se diferenças da ordem de 10^{-2} entre as técnicas com melhor desempenho, isto é, diferenças da ordem de centésimos percentuais.

As ameaças relacionadas à validade de construção dizem respeito a eventos que podem impedir que a configuração do experimento reflita adequadamente a construção do relacionamento de causa e efeito (entre o tratamento e o resultado) em estudo. Em outras palavras, são eventos que podem prejudicar a medição correta no estudo. Uma das ameaças dessa categoria existe porque as técnicas do tipo Dinâmico Local demandam que as séries de entrada possuam um tamanho muito maior do que demandam as demais técnicas. Por exemplo, para a realização de um experimento com esse tipo de técnica neste trabalho, foi

necessário que todas as séries de entrada tivessem ao menos 25 elementos. Essa restrição existe porque as técnicas as técnicas do tipo Dinâmico Local, assim como todas as outras, necessitam de 10 elementos separados para comporem a série de valores efetivos e de um número mínimo de elementos na série de entrada requerido pelas técnicas que agrega (aproximadamente 5 elementos). Somam-se a essa quantidade mais 10 elementos que elas utilizam como valores efetivos para encontrar a melhor instância de técnica de predição para utilizar com a entrada, totalizando 25 elementos. Em contraste, além dos 10 elementos separados para comporem a série de valores efetivos, as demais técnicas demandam aproximadamente 5 elementos, totalizando 15 elementos. Dessa forma, a técnica Dinâmico Local impossibilitou as demais técnicas de explorar grande parte da informação histórica contida nas séries (aproximadamente 10 elementos), prejudicando principalmente as técnicas da família KNN, que dependem da identificação de séries similares. Outra ameaça é devido ao fato de as medidas que compõem as séries compreendem todo o histórico do projeto, de seu primeiro *commit* no Sistema de Controle de Versões até o último registrado antes do dia do lançamento da versão 1.7 da linguagem Java, como foi explicado na Seção 5.2. Não se considerou, portanto, a eficácia das técnicas em diferentes momentos da vida do software, o que pode gerar resultados diferentes, já que os valores das séries utilizados para a predição pelas técnicas apresentam outros padrões de evolução, como já foi levantado na Seção 5.5.

5.7 CONSIDERAÇÕES FINAIS

Neste capítulo, os resultados dos experimentos com as técnicas descritas no Capítulo 4 foram reunidos e avaliados de acordo com seu valor de ERM. Esses resultados foram agrupados pela base de medidas de onde as séries obtiveram seus valores, isto é, por métrica e por projeto. Dentro de cada agrupamento, os resultados foram classificados pela técnica utilizada no experimento e, em seguida, pelo valor de ERM. Por exemplo, os resultados dos experimentos com KNN foram agrupados e, em seguida, cada instância da técnica foi classificada pelo seu valor de ERM. O menor ERM de cada técnica, em cada horizonte, foi obtido e registrado nas tabelas mostradas na Seção 5.5. Além desses valores, também foi incluída uma coluna, denominada MELHOR, que contém a contagem do número de vezes que cada técnica teve o melhor desempenho ao longo dos 10 horizontes.

Diante dos resultados dos experimentos, observou-se, pela análise de MELHOR, que as técnicas Regressão Linear, Regressão Quadrática, Dinâmico Local e KNN não possuem grande poder preditivo no contexto das métricas e dos projetos em que foram aplicadas. Em

contraste, KNN-IR, KNN-IA e Médias Móveis são as que apresentam maior poder preditivo, já que figuram como campeãs em vários projetos ao longo dos horizontes.

Os resultados evidenciam uma predominância de KNN-IR e KNN-IA sobre as demais técnicas. Percebe-se que essas técnicas se adequaram a diferentes contextos, isto é, a diferentes projetos e diferentes tipos de dados (métricas). No entanto, como discutido na Seção 5.6, não é possível fundamentar conclusões sobre o contexto para o qual cada técnica apresenta os melhores desempenhos. Por exemplo, não é possível estabelecer que determinada técnica é a melhor escolha para realizar predições com séries compostas de valores de determinada métrica ou obtidos de determinado projeto. Os valores de ERM diferem em alguns poucos centésimos, em geral devido ao melhor desempenho de determinada técnica sobre outra na predição de um pequeno conjunto de séries específicas. Em alguns casos, a vitória de uma técnica sobre outra foi devido não à sua capacidade preditiva, mas a um fator aleatório, como a ocorrência de uma queda brusca no valor da métrica. São necessários estudos adicionais, como aqueles que serão elencados como trabalhos futuros no Capítulo 6, para obter um conhecimento mais aprofundado sobre o comportamento de cada técnica diante dos dados com os quais realiza predições.

CAPÍTULO 6 – CONCLUSÃO

6.1 INTRODUÇÃO

A predição no contexto do desenvolvimento de software traz muitos benefícios ao projeto que a utiliza, permitindo que a equipe trabalhe preventivamente ao invés de reativamente. Métricas são utilizadas com o objetivo de oferecer uma percepção do estado atual do software, entretanto ao sinalizarem algum problema, ele já está presente no software, o que pode levar um projeto a uma situação de alta instabilidade. Na literatura, há muitos trabalhos que objetivam estudar e propor soluções relacionadas à predição através de métricas estruturais, como Número de Linhas de Código, Complexidade Ciclomática e Número de Métodos Públicos. Entretanto, atendendo a uma demanda ainda não contemplada por essas abordagens, após a realização de uma Revisão Sistemática da Literatura no Capítulo 3, o presente trabalho tem como objetivo apresentar um *framework* que auxilia a realização de experimentos e a construção de aplicações destinadas a realizar predições.

O *framework* apresentado no Capítulo 4, denominado Peixe-Dipnoico, oferece entidades que permitem o gerenciamento da execução de experimentos com vários tipos de métricas e em vários projetos. Além disso, permitem a representação da evolução dos valores de métricas obtidas de uma classe ao longo de suas revisões através de séries temporais, a obtenção de dados de diferentes fontes (Sistemas de Controle de Versões) e a exibição das saídas dos experimentos e das predições sob diferentes formatos.

Utilizando os recursos oferecidos pelo *framework*, o Capítulo 5 apresentou os resultados de experimentos realizados com séries temporais compostas de valores de métricas extraídas de projetos *open-source* de grande relevância e popularidade. Nesses experimentos, a eficácia de técnicas novas e tradicionais de predição foi comparada. A análise dos resultados evidencia uma predominância das técnicas KNN-IR e KNN-IA sobre as demais técnicas. Percebeu-se que essas técnicas se adequaram a diferentes contextos, isto é, a diferentes projetos e diferentes tipos de dados (métricas). Não foi possível fundamentar conclusões sobre o contexto para o qual cada técnica apresenta os melhores desempenhos, visto que os valores de ERM encontrados nos experimentos diferem em alguns poucos centésimos, em geral devido ao melhor desempenho de determinada técnica sobre outra na predição de um pequeno conjunto de séries específicas.

6.2 CONTRIBUIÇÕES

Este trabalho deixa as seguintes contribuições:

- Um *framework* que auxilia a realização de experimentos e a construção de aplicações destinadas a realizar predições.
- Uma Revisão Sistemática da Literatura, que faz um levantamento das abordagens que realizam predições utilizando medidas estruturais de software.
- Um conjunto de experimentos que compara o desempenho tanto de técnicas novas, implementadas no contexto deste trabalho, quanto de técnicas tradicionais de predição, o que permitiu o levantamento de algumas evidências sobre seu funcionamento.
- Protótipos de aplicações para o gerenciamento de experimentos com os algoritmos de predição, desenvolvidos com a utilização do *framework*, com os quais foram realizados os experimentos apresentados no Capítulo 5.

6.3 LIMITAÇÕES

Uma das limitações desta abordagem reside no fato de que durante a modelagem da evolução das medidas, somente são considerados valores. Em outras palavras, a série temporal contém apenas valores numéricos, o que não é suficiente para representar a realidade do desenvolvimento. Nessa realidade, os valores das métricas mudam devido a fatores como aqueles relacionados à ação humana (cansaço, imperícia, desatenção, etc.) e relacionados ao projeto (tempo, requisitos novos, etc.). Considerar não só as métricas de produto (as métricas estruturais), mas também as métricas de processo, de gestão, de pessoas e outras, bem como o relacionamento entre elas, pode trazer novas perspectivas que ajudem a melhorar a modelagem da evolução das métricas estruturais.

Outra limitação está relacionada à granularidade das medições. A abordagem obtém e utiliza medidas extraídas de classes. Entretanto, muitas vezes as modificações se concentram em determinado método de uma classe. Nesses casos, com a granularidade em nível de método seria possível estabelecer modelos de predição mais precisos, pois seriam removidas as interferências causadas por outros métodos. Além da granularidade das medições, há também a granularidade referente ao estilo do desenvolvedor, como explicado no Capítulo 5, no qual o *commit* de um desenvolvedor nem sempre equivale, em termos de valor agregado, ao *commit* de outro.

6.4 TRABALHOS FUTUROS

Com base nas limitações listadas neste capítulo e nas ameaças à validade apresentadas no Capítulo 5, podem ser identificadas as seguintes oportunidades de pesquisa sobre o tema abordado neste trabalho:

- Incluir no *framework* a modelagem da ação de fatores que podem interferir variação dos valores de métricas, como ação humana, por exemplo, de modo a melhorar o desempenho das predições.
- Implementar o suporte à predição utilizando valores em variados níveis de granularidade.
- Aplicar o *framework* na predição de anomalias de software, utilizando como base o trabalho de MACIA (2012), conforme descrito no Capítulo 4.
- Utilizando novos projetos e mais métricas, realizar um estudo mais aprofundado de modo a obter conclusões mais precisas a respeito do desempenho e da adequação das técnicas em face a diferentes contextos, como nível de maturidade e/ou tamanho do projeto, métrica utilizada, comportamento apresentado pelos valores das séries, dentre outros.
- Expandir as funcionalidades do *framework*, de modo a torná-lo habilitado a lidar com diferentes demandas relacionadas à predição de medidas de software.

REFERÊNCIAS

AGGARWAL, C. C.; HINNEBURG, A.; KEIM, D. A. On the Surprising Behavior of Distance Metrics in High Dimensional Spaces. In: INTERNATIONAL CONFERENCE ON DATABASE THEORY, ICDT '01, London, UK, UK: Springer-Verlag, 2001. 420–434 p.

BAISCH, E.; LIEDTKE, T. Comparison of conventional approaches and soft-computing approaches for software quality prediction. In: INTERNATIONAL CONFERENCE ON SYSTEMS, MAN, AND CYBERNETICS, Orlando, FL, USA: IEEE Computer Society, 1997. 1045–1049 p. v. 2.

BARRETO, A. ***Uma abordagem para definição de processos baseada em reutilização visando à alta maturidade em processos***. Tese de Doutorado. Rio de Janeiro, Brasil: Universidade Federal do Rio de Janeiro, 2011

BHAT, T.; NAGAPPAN, N. Tempest: Towards early identification of failure-prone binaries. In: INTERNATIONAL CONFERENCE ON DEPENDABLE SYSTEMS AND NETWORKS WITH FTCS AND DCC, Anchorage, AK: IEEE Computer Society, 2008. 116–121 p.

BOSCH, J. *Design and Use of Software Architectures: Adopting and Evolving a Product-Line Approach*. [S.l.]: Addison-Wesley Professional, 2000.

BRIAND, L. C.; WÜST, J. The Impact of Design Properties on Development Cost in Object-Oriented Systems. In: INTERNATIONAL SYMPOSIUM ON SOFTWARE METRICS, METRICS '01, Washington, DC, USA: IEEE Computer Society, 2001. 260 p.

BROCKWELL, P. J.; DAVIS, R. A. *Introduction to Time Series and Forecasting*. 2. ed. USA: Springer, 2002.

CAPILUPPI, A.; FERNANDEZ-RAMIL, J. A model to predict anti-regressive effort in Open Source Software. In: INTERNATIONAL CONFERENCE ON SOFTWARE MAINTENANCE, Paris, France: IEEE Computer Society, out. 2007. 194–203 p.

CATAL, C.; DIRI, B.; OZUMUT, B. An Artificial Immune System Approach for Fault Prediction in Object-Oriented Software. In: INTERNATIONAL CONFERENCE ON DEPENDABILITY OF COMPUTER SYSTEMS, Szklarska: IEEE Computer Society, 2007. 238–245 p.

CHEATHAM, T. J.; YOO, J. P.; WAHL, N. J. Software testing: a machine learning experiment. In: ACM ANNUAL COMPUTER SCIENCE CONFERENCE, CSC '95, New York, NY, USA: ACM, 1995. 135–141 p.

CHIDAMBER, S. R.; KEMERER, C. F. A metrics suite for object oriented design. *IEEE Transactions on Software Engineering*, 1994. v. 20.

CLÁUDIO, D. M.; MARINS, J. M. *Cálculo Numérico Computacional - Teoria e Prática*. 2. ed. São Paulo: Atlas, 1994.

DANIEL, B.; BOSHERNITSAN, M. Predicting Effectiveness of Automatic Testing Tools. In: INTERNATIONAL CONFERENCE ON AUTOMATED SOFTWARE ENGINEERING, L'Aquila: IEEE Computer Society, 2008. 363–366 p.

DAVIDSSON, M.; ZHENG, J.; NAGAPPAN, N.; WILLIAMS, L.; VOUK, M. GERT: an

empirical reliability estimation and testing feedback tool. In: INTERNATIONAL SYMPOSIUM ON SOFTWARE RELIABILITY ENGINEERING, Saint-Malo, Bretagne, France: IEEE Computer Society, 2004. 269–280 p.

EHLERS, R. S. *Análise de Séries Temporais*. , 2009

FERRERO, C. A. ***Algoritmo kNN para previsão de dados temporais: funções de previsão e critérios de seleção de vizinhos próximos aplicados a variáveis ambientais em limnologia***. Dissertação de Mestrado. São Paulo, SP, Brasil: Universidade de São Paulo, 2009

FERZUND, J.; AHSAN, S. N.; WOTAWA, F. Software change classification using hunk metrics. In: INTERNATIONAL CONFERENCE ON SOFTWARE MAINTENANCE, Edmonton, AB: IEEE Computer Society, 2009. 471–474 p.

GEGICK, M.; WILLIAMS, L.; OSBORNE, J.; VOUK, M. Prioritizing software security fortification through code-level metrics. In: WORKSHOP ON QUALITY OF PROTECTION, QoP '08, New York, NY, USA: ACM, 2008. 31–38 p.

GRAY, D.; BOWES, D.; DAVEY, N.; SUN, Y.; CHRISTIANSON, B. Using the Support Vector Machine as a Classification Method for Software Defect Prediction with Static Code Metrics. Em: PALMER-BROWN, D.; DRAGANOVA, C.; PIMENIDIS, E.; MOURATIDIS, H. (Org.). *Engineering Applications of Neural Networks*. Communications in Computer and Information Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. v. 43. p. 223–234.

GUJARATI, D. N. *Basic Econometrics*. 4. ed. New York, NY, USA: McGraw-Hill, 2003.

GUO, P.; LYU, M. R. Software quality prediction using mixture models with EM algorithm. In: CONFERENCE ON QUALITY SOFTWARE, Honk Kong: IEEE Computer Society, 2000. 69–78 p.

HADERER, N.; KHOMH, F.; ANTONIOL, G. SQUANER: A framework for monitoring the quality of software systems. In: INTERNATIONAL CONFERENCE ON SOFTWARE MAINTENANCE, Timisoara, Romania: IEEE Computer Society, 2010. 1–4 p.

HATA, H.; MIZUNO, O.; KIKUNO, T. Historage: fine-grained version control system for Java. In: INTERNATIONAL WORKSHOP ON THE PRINCIPLES OF SOFTWARE EVOLUTION, IWPSE-EVOL '11, Szeged, Hungary: ACM New York, 2011. 96–100 p.

IEEE. *Std 610.12 - IEEE Standard Glossary of Software Engineering Terminology*. . New York, NY, USA: Institute of Electrical and Electronics Engineers, 1990.

JALBERT, K.; BRADBURY, J. S. Predicting mutation score using source code and test suite metrics. In: INTERNATIONAL WORKSHOP ON REALIZING ARTIFICIAL INTELLIGENCE SYNERGIES IN SOFTWARE ENGINEERING, Zurich, Switzerland: IEEE Computer Society, 2012. 42–46 p.

KAMIYA, T.; KUSUMOTO, S.; INOUE, K. Prediction of fault-proneness at early phase in object-oriented development. In: IEEE INTERNATIONAL SYMPOSIUM ON OBJECTORIENTED REAL-TIME DISTRIBUTED COMPUTING, Saint-Malo, Bretagne, France: [s.n.], 1999. 253–258 p.

KAN, S. H. *Metrics and Models in Software Quality Engineering*. 2. ed. [S.l.]: Addison-Wesley, 2002.

KAUR, A.; BRAR, A. S.; SANDHU, P. S. An empirical approach for software fault prediction. In: INTERNATIONAL CONFERENCE ON INDUSTRIAL AND

INFORMATION SYSTEMS, Mangalore, India: [s.n.], 2010. 261–265 p.

KAUR, A.; SANDHU, P. S.; BRA, A. S. Early Software Fault Prediction Using Real Time Defect Data. In: INTERNATIONAL CONFERENCE ON MACHINE VISION, Dubai: IEEE Computer Society, 2009. 242–245 p.

KAUR, D.; KAUR, A.; GULATI, S.; AGGARWAL, M. A clustering algorithm for software fault prediction. In: INTERNATIONAL CONFERENCE ON COMPUTER AND COMMUNICATION TECHNOLOGY, Allahabad, Uttar Pradesh: IEEE Computer Society, 2010. 603–607 p.

KHOSHGOFTAAR, T.M.; ALLEN, E. B.; XU, Z. Predicting testability of program modules using a neural network. In: SYMPOSIUM ON APPLICATION-SPECIFIC SYSTEMS AND SOFTWARE ENGINEERING TECHNOLOGY, Richardson, TX: IEEE Computer Society, 2000. 57–62 p.

KHOSHGOFTAAR, T.M.; MUNSON, J. C.; LANNING, D. L. A comparative study of predictive models for program changes during system testing and maintenance. In: CONFERENCE ON SOFTWARE MAINTENANCE, Montreal, Canada: IEEE Computer Society, 1993. 72–79 p.

KHOSHGOFTAAR, T.M.; PANDYA, A. S.; MORE, H. B. A neural network approach for predicting software development faults. In: INTERNATIONAL SYMPOSIUM ON SOFTWARE RELIABILITY ENGINEERING, Research Triangle Park, NC: IEEE Computer Society, 1992. 83–89 p.

KHOSHGOFTAAR, T.M.; SZABO, R. M. Improving code churn predictions during the system test and maintenance phases. In: INTERNATIONAL CONFERENCE ON SOFTWARE MAINTENANCE, Victoria, BC, Canada: IEEE Computer Society, 1994a. 58–67 p.

KHOSHGOFTAAR, T.M.; SZABO, R. M. Improving neural network predictions of software quality using principal components analysis. In: INTERNATIONAL CONFERENCE ON NEURAL NETWORKS, Orlando, FL, USA: IEEE Computer Society, 1994b. 3295–3300 p. v. 5.

KHOSHGOFTAAR, TAGHI M.; ALLEN, E. B.; JONES, W. D.; HUDEPOHL, J. P. Data Mining for Predictors of Software Quality. *International Journal of Software Engineering and Knowledge Engineering*, out. 1999. v. 09.

KHOSHGOFTAAR, TAGHI M.; SZABO, R. M. Investigating ARIMA models of software system quality. *Software Quality Journal*, 1 mar. 1995. v. 4.

KUMAR, R.; RAI, S.; TRAHAN, J. L. Neural-network techniques for software-quality evaluation. In: ANNUAL RELIABILITY AND MAINTAINABILITY SYMPOSIUM, Anaheim, CA, USA: IEEE Computer Society, 1998. 155–161 p.

KUTLUBAY, O.; TURHAN, B.; BENER, A. B. A Two-Step Model for Defect Density Estimation. In: CONFERENCE ON SOFTWARE ENGINEERING AND ADVANCED APPLICATIONS, Lubeck, Germany: IEEE Computer Society, 2007. 322–332 p.

LI, L.; LEUNG, H. Mining Static Code Metrics for a Robust Prediction of Software Defect-Proneness. In: INTERNATIONAL SYMPOSIUM ON EMPIRICAL SOFTWARE ENGINEERING AND MEASUREMENT, Banff, AB, Canada: IEEE Computer Society, 2011. 207–214 p.

MACIA, I.; ARCOVERDE, R.; GARCIA, A.; CHAVEZ, C.; VON STAA, A. On the Relevance of Code Anomalies for Identifying Architecture Degradation Symptoms. In:

CONFERENCE ON SOFTWARE MAINTENANCE AND REENGINEERING, CSMR ’12, Washington, DC, USA: IEEE Computer Society, 2012. 277–286 p.

MACIA, I.; GARCIA, J.; POPESCU, D.; GARCIA, A.; MEDVIDOVIC, N.; VON STAA, A. Are automatically-detected code anomalies relevant to architectural modularity? An exploratory analysis of evolving systems. In: INTERNATION CONFERENCE ON ASPECT-ORIENTED SOFTWARE DEVELOPMENT, AOSD ’12, New York, NY, USA: ACM, 2012. 167–178 p.

MAIMON, O.; ROKACH, L. *Data Mining and Knowledge Discovery Handbook*. 2. ed. New York, USA: Springer, 2010.

MARCHETTO, A.; TRENTINI, A. Evaluating Web applications testability by combining metrics and analogies. In: INTERNATIONAL CONFERENCE ON INFORMATION AND COMMUNICATIONS TECHNOLOGY, Cairo: IEEE Computer Society, 2005. 751–779 p.

MCCABE, T. J. A Complexity Measure. *IEEE Trans. Softw. Eng.*, 1976. v. 2.

MEIRELLES, P. R. M. *Levantamento de Métricas de Avaliação de Projetos de Software Livre*. Relatório Técnico. São Paulo, SP, Brasil: Universidade de São Paulo, dez. 2008.

MONTONI, M. ***Uma Abordagem para Condução de Iniciativas de Melhoria de Processos de Software***. Tese de Doutorado. Rio de Janeiro, Brasil: Universidade Federal do Rio de Janeiro, 2007

MORETTIN, P. A.; TOLOI, C. M. DE C. *Modelos para previsão de séries temporais*. Rio de Janeiro: Instituto de Matemática Pura e Aplicada, 1981.

MUELLER, A. ***Uma aplicação de Redes Neurais artificiais na previsão do mercado acionário***. Dissertação de Mestrado. Florianópolis, SC, Brasil: Universidade Federal de Santa Catarina, 1996

PIATTINI, M.; GENERO, M.; JIMÉNEZ, L. A Metric-Based Approach for Predicting Conceptual Data Models Maintainability. *International Journal of Software Engineering and Knowledge Engineering*, 2001. v. 11.

PIGHIN, M.; MARZONA, A. Reducing Corrective Maintenance Effort Considering Module’s History. In: CONFERENCE ON SOFTWARE MAINTENANCE AND REENGINEERING, CSMR ’05, Washington, DC, USA: IEEE Computer Society, 2005. 232–235 p.

PIZZI, N. J.; PEDRYCZ, W. Predicting Qualitative Assessments Using Fuzzy Aggregation. In: NORTH AMERICA FUZZY INFORMATION PROCESSING SOCIETY, Montreal, Canada: IEEE Computer Society, 2006. 267–272 p.

PREDICTION. PREDICTION. *Wikipedia, the free encyclopedia*. , 15 jun. 2013. Disponível em: <https://en.wikipedia.org/w/index.php?title=Prediction&oldid=559976025>. Acesso em 16 jun. 2013.

PRESSMAN, R. S. *Software Engineering: A Practitioner’s Approach*. 6th. ed. USA: McGraw-Hill, 2005.

QUAH, T. S.; THWIN, M. M. T. Utilizing Computational Intelligence in Estimating Software Readiness. In: INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS, Vancouver, BC: IEEE Computer Society, 2006. 2999–3006 p.

SAHRAOUI, H. A.; BOUKADOUM, A. M.; LOUNIS, H.; ETHEVE, F. Predicting class libraries interface evolution: an investigation into machine learning approaches. In:

SOFTWARE ENGINEERING CONFERENCE, Singapore, China: IEEE Computer Society, 2000. 456–464 p.

SAMI, A.; FAKHRAHMAD, S. M. Design-level metrics estimation based on code metrics. In: SYMPOSIUM ON APPLIED COMPUTING, SAC '10, New York, NY, USA: ACM, 2010. 2531–2535 p.

SANDHU, P. S.; GOEL, R.; BRAR, A. S.; KAUR, J.; ANAND, S. A model for early prediction of faults in software systems. In: INTERNATIONAL CONFERENCE ON COMPUTER AND AUTOMATION ENGINEERING, Singapore, China: IEEE Computer Society, 2010. 281–285 p. v. 4.

SANDHU, P. S.; KAUR, M.; KAUR, A. A Density Based Clustering approach for early detection of fault prone modules. In: INTERNATIONAL CONFERENCE ON ELECTRONICS AND INFORMATION ENGINEERING, Kyoto, Japan: IEEE Computer Society, 2010. V2–525–V2–530 p. v. 2.

SHARAFAT, A. R.; TAHVILDARI, L. A Probabilistic Approach to Predict Changes in Object-Oriented Software Systems. In: CONFERENCE ON SOFTWARE MAINTENANCE AND REENGINEERING, CSMR '07, Washington, DC, USA: IEEE Computer Society, 2007. 27–38 p.

SHAW, W.H., J.; HOWATT, J. W.; MANESS, R. S.; MILLER, D. M. A software science model of compile time. *IEEE Transactions on Software Engineering*, 1989. v. 15.

SHIN, Y.; BELL, R.; OSTRAND, T.; WEYUKER, E. Does calling structure information improve the accuracy of fault prediction? In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, Vancouver, Canada: IEEE Computer Society Washington, 16 maio 2009. 61–70 p.

SHIN, YONGHEE; BELL, R. M.; OSTRAND, T. J.; WEYUKER, E. J. On the use of calling structure information to improve fault prediction. *Empirical Software Engineering*, 1 ago. 2012. v. 17.

SOLIGEN, R. VAN; BERGHOUT, E. *The Goal/Question/Metric Method: a practical guide for quality improvement of software development*. Cambridge, United Kingdom: McGraw-Hill, 1999.

S-PLUS 7.0 Guide to Statistics. [S.l.]: Insightful Corporation, abr. 2005

TAKAHASHI, M.; MIYAKE, T.; HANATA, S. Statistically-based program size estimation. In: ANNUAL INTERNATIONAL COMPUTER SOFTWARE AND APPLICATIONS CONFERENCE, Orlando, FL, USA: IEEE Computer Society, 1989. 574–579 p.

TAN, P.-N. *Introduction to Data Mining*. [S.l.]: Pearson Addison Wesley, 2006.

TARVO, A. Using Statistical Models to Predict Software Regressions. In: INTERNATIONAL SYMPOSIUM ON SOFTWARE RELIABILITY ENGINEERING, ISSRE '08, Washington, DC, USA: IEEE Computer Society, 2008. 259–264 p.

THWIN, M. M. T.; QUAH, T.-S. Application of neural network for predicting software development faults using object-oriented design metrics. In: INTERNATIONAL CONFERENCE ON NEURAL INFORMATION PROCESSING, Singapore, China: IEEE Computer Society, 2002. 2312–2316 p. v. 5.

TIAN, Y.; CHEN, C.; ZHANG, C. AODE for Source Code Metrics for Improved Software Maintainability. In: INTERNATIONAL CONFERENCE ON SEMANTICS, KNOWLEDGE AND GRID, Beijing, China: IEEE Computer Society, 2008. 330–335 p.

TONU, S. A.; ASHKAN, A.; TAHVILDARI, L. Evaluating architectural stability using a metric-based approach. In: EUROPEAN CONFERENCE ON SOFTWARE MAINTENANCE AND REENGINEERING, Tahiti, Polinésia Francesa: IEEE Computer Society, 2006. 10 pp.–270 p.

TOSUN, A.; TURHAN, B.; BENER, A. Practical considerations in deploying AI for defect prediction: a case study within the Turkish telecommunication industry. In: INTERNATIONAL CONFERENCE ON PREDICTOR MODELS IN SOFTWARE ENGINEERING, PROMISE '09, New York, NY, USA: ACM, 2009a. 11:1–11:9 p.

TOSUN, A.; TURHAN, B.; BENER, A. Validation of network measures as indicators of defective modules in software systems. In: INTERNATIONAL CONFERENCE ON PREDICTOR MODELS IN SOFTWARE ENGINEERING, PROMISE '09, New York, NY, USA: ACM, 2009b. 5:1–5:9 p.

TOTH, G.; VEGH, A. Z.; BESZEDES, A.; GYIMOTHY, T. Adding Process Metrics to Enhance Modification Complexity Prediction. In: INTERNATIONAL CONFERENCE ON PROGRAM COMPREHENSION, Kingston, Canada: IEEE Computer Society, 2011. 201–204 p.

ÜBERHUBER, C. W. *Numerical Computation Bd. 1.: Methods, Software, and Analysis.* Berlin, Germany: Springer, 1997.

WANG, Y.; SMITH, M. Release date prediction for telecommunication software using Bayesian Belief Networks. In: CANADIAN CONFERENCE ON ELECTRICAL AND COMPUTER ENGINEERING, Winnipeg, Manitoba, Canada: IEEE Computer Society, 2002. 738–742 p. v. 2.

WOHLIN, C.; RUNESON, P.; HÖST, M.; OHLSSON, M. C.; REGNELL, B.; WESSLÉN, A. *Experimentation in software engineering: an introduction*. Norwell, MA, USA: Kluwer Academic Publishers, 2000.

XING, F.; GUO, P.; LYU, M. R. A novel method for early software quality prediction based on support vector machine. In: INTERNATIONAL SYMPOSIUM ON SOFTWARE RELIABILITY ENGINEERING, Chicago, IL, USA: IEEE Computer Society Washington, 2005. 213–222 p.

APÊNDICE A – DADOS COLETADOS DAS PUBLICAÇÕES SELECIONADAS

Dados da Publicação	
Título	A Model For Early Prediction Of Faults In Software Systems
Autores	Sandhu P.S., Brar A.S., Goel R., Kaur J., Anand S.
Ano da publicação	2010
Conferência	The 2nd International Conference on Computer and Automation Engineering, ICCAE 2010
Resumo	
A abordagem investiga se a combinação de métricas disponíveis em etapas iniciais do ciclo de vida do software (i.e., métricas de requisitos) com métricas disponíveis em etapas posteriores (i.e., métricas de código) podem ser usadas para identificar módulos suscetíveis a falha. Também traz à tona a possibilidade de haver casos em que o atributo de classificação (i.e., informação se o artefato de software tem defeito ou não) não pode ser obtido. Para esses casos, a abordagem realiza um pré-processamento para descobrir a classe dos módulos através do algoritmo de clusterização *K-Means*. Obtido o atributo de classificação das instâncias da base de dados, utiliza-se o algoritmo de classificação C4.5 para realizar a predição.	
QS1: De onde são obtidos os dados para a predição?	
Bases de medidas: NASA MDP	
QS2: Qual a granularidade dos artefatos de software analisados para construir a predição?	
Módulo	
QS3: Que tipos de características estruturais são usados?	
Métricas de Código	
QS4: Para que propósito a abordagem é destinada?	
Predição de Falhas: *Fault-proneness*	
QS5: Qual a técnica utilizada para construir a predição?	
Categoria: Classificação Nome: C4.5 Descrição: A técnica proposta realiza uma clusterização (*K-means*) para descobrir a classe das instâncias da base de dados. Descobrindo essas instâncias, utiliza o algoritmo de classificação C4.5.	
QS6: Quais métricas de desempenho foram utilizadas para avaliar a abordagem?	
Cobertura, Taxa de Falsos Positivos	
QS7: Qual é o desempenho aproximado das predições fornecido pela abordagem?	
Cobertura = 100%	
QS8: Oferece apoio ferramental?	
Não	

Dados da Publicação	
Título	Gert: An Empirical Reliability Estimation And Testing Feedback Tool
Autores	Davidsson, M.; Zheng, J.; Nachiappan Nagappan; Williams, L.; Vouk, M.
Ano da publicação	2004
Conferência	Software Reliability Engineering, 15th International Symposium on
Resumo	
O artigo apresenta uma ferramenta que calcula estimativas sobre a confiabilidade do software e quantifica o intervalo de confiança da estimativa. Para o cálculo das estimativas, utiliza a técnica de regressão *Multivariate Regression*.	
QS1: De onde são obtidos os dados para a predição?	
Projeto de software: Arquivos de código-fonte abertos no Eclipse	
QS2: Qual a granularidade dos artefatos de software analisados para construir a predição?	

Classe
QS3: Que tipos de características estruturais são usados?
Métricas de Código: Número de Linhas de Código de Teste, Número de Linhas de Código, etc
QS4: Para que propósito a abordagem é destinada?
Predição de Qualidade: Confiabilidade do software
QS5: Qual a técnica utilizada para construir a predição?
Categoria: Regressão Nome: *Multivariate Regression*
QS6: Quais métricas de desempenho foram utilizadas para avaliar a abordagem?
Não houve validação experimental
QS7: Qual é o desempenho aproximado das predições fornecido pela abordagem?
Não houve validação experimental
QS8: Oferece apoio ferramental?
Sim

Dados da Publicação	
Título	A Novel Method For Early Software Quality Prediction Based On Support Vector Machine
Autores	Fei Xing; Ping Guo; Lyu, M.R.
Ano da publicação	2005
Conferência	Software Reliability Engineering, 16th IEEE International Symposium on
Resumo	
O artigo propõe um modelo de classificação baseado em SVM (*Support Vector Machine*) para realizar predições sobre a propensão à ocorrência de falhas (*fault-proneness*).	
QS1: De onde são obtidos os dados para a predição?	
Projeto de software: software de processamento de imagens médicas	
QS2: Qual a granularidade dos artefatos de software analisados para construir a predição?	
Módulo	
QS3: Que tipos de características estruturais são usados?	
Métricas de Código: Número de linhas de Código, Complexidade Ciclomática, Tamanho do programa (Halstead e Jensen), etc	
QS4: Para que propósito a abordagem é destinada?	
Predição de Falhas: *Fault-proneness*	
QS5: Qual a técnica utilizada para construir a predição?	
Categoria: Classificação Nome: *Transductive Support Vector Machine* (TSVM)	
QS6: Quais métricas de desempenho foram utilizadas para avaliar a abordagem?	
Acurácia	
QS7: Qual é o desempenho aproximado das predições fornecido pela abordagem?	
Acurácia = 90.03%	
QS8: Oferece apoio ferramental?	
Não	

Dados da Publicação	
Título	A Metric-Based Approach For Predicting Conceptual Data Models Maintainability
Autores	Piattini M., Genero M., Jimenez L.
Ano da publicação	2001
Conferência	International Journal of Software Engineering and Knowledge Engineering
Resumo	
O artigo propõe métricas para avaliar a complexidade interna de Diagramas de Entidade-Relacionamento (DER) e as utiliza para compor o	

modelo de predição proposto. Esse modelo é baseado em um sistema de regras *fuzzy* denominado *Fuzzy Classification and Regression Tree* (FCART).
QS1: De onde são obtidos os dados para a predição?
Diagrama: Diagramas Entidade-Relacionamento
QS2: Qual a granularidade dos artefatos de software analisados para construir a predição?
Entidade de diagrama
QS3: Que tipos de características estruturais são usados?
Métricas de Diagramas: Número de Entidades, Número de Atributos, Número de Relacionamentos, etc
QS4: Para que propósito a abordagem é destinada?
Predição de Qualidade: Manutenibilidade
QS5: Qual a técnica utilizada para construir a predição?
Categoria: Híbrido de Classificação e Regressão Nome: *Fuzzy Classification and Regression Tree* (FCART)
QS6: Quais métricas de desempenho foram utilizadas para avaliar a abordagem?
Não houve validação experimental
QS7: Qual é o desempenho aproximado das predições fornecido pela abordagem?
Não houve validação experimental
QS8: Oferece apoio ferramental?
Não

Dados da Publicação	
Título	An Artificial Immune System Approach For Fault Prediction In Object-Oriented Software
Autores	Catal C., Diri B., Ozumut B.
Ano da publicação	2007
Conferência	Proceedings of International Conference on Dependability of Computer Systems
Resumo	
O artigo mostra uma evidência de relação entre propensão a falhas (*fault-proneness*) e métricas de *Chidamber-Kemerer* e propõe um modelo de predição, o *Artificial Immune Recognition Systems*, utilizando métricas do paradigma orientado a objetos.	
QS1: De onde são obtidos os dados para a predição?	
Bases de medidas: NASA PROMISE	
QS2: Qual a granularidade dos artefatos de software analisados para construir a predição?	
Classe, Método	
QS3: Que tipos de características estruturais são usados?	
Métricas OO: Weighted Methods per Class, Depth of Inheritance Tree, Percentage of Public/Protected Data Métricas de Dependência: Fan-In, Lack of Cohesion in Methods	
QS4: Para que propósito a abordagem é destinada?	
Predição de Falhas: *Fault-proneness*	
QS5: Qual a técnica utilizada para construir a predição?	
Categoria: Classificação Nome: *Artificial Immune Recognition Systems* (AIRS)	
QS6: Quais métricas de desempenho foram utilizadas para avaliar a abordagem?	
Precisão, Cobertura, Acurácia Outros: Taxa de Verdadeiros Negativos	
QS7: Qual é o desempenho aproximado das predições fornecido pela abordagem?	
Acurácia = 72.4%	
QS8: Oferece apoio ferramental?	
Não	

Dados da Publicação	
Título	On The Use Of Calling Structure Information To Improve Fault Prediction
Autores	Shin Y., Bell R.M., Ostrand T.J., Weyuker E.J.
Ano da publicação	2012
Conferência	Empirical Software Engineering

Resumo

O artigo propõe um modelo para predição de falhas que utiliza atributos de "Calling Structure" combinadas com métricas de código comumente usadas. Esse modelo é uma metodologia que utiliza o algoritmo de regressão *Negative Binomial Regression*.

QS1: De onde são obtidos os dados para a predição?

Projeto de software

QS2: Qual a granularidade dos artefatos de software analisados para construir a predição?

Classe, Método

QS3: Que tipos de características estruturais são usados?

Métricas de Código: Número de Linhas de Código

Específicas: Medidas das "Calling Structures"

QS4: Para que propósito a abordagem é destinada?

Predição de Falhas

QS5: Qual a técnica utilizada para construir a predição?

Categoria: Técnica Específica

Descrição: A técnica *Negative Binomial Regression* é utilizada para calcular o número estimado de falhas das classes. As classes são ordenadas em ordem decrescente do tamanho estimado para cada uma delas e a porcentagem de falhas corretamente preditas é calculada para as primeiras 20% da lista ordenada. Para predizer falhas na Release N, constroem-se modelos usando os dados obtidos da Release 1 até a Release N-1. Então, aplica-se o modelo aos dados coletados para a Release N.

QS6: Quais métricas de desempenho foram utilizadas para avaliar a abordagem?

Avaliação específica: As estimativas foram ordenadas em ordem decrescente e foi calculada a Acurácia dos elementos do topo dessa lista (20% do total).

QS7: Qual é o desempenho aproximado das predições fornecido pela abordagem?

Avaliação específica: Acurácia = 78%

QS8: Oferece apoio ferramental?

Não

Dados da Publicação	
Título	A Clustering Algorithm For Software Fault Prediction
Autores	Kaur, D.; Kaur, A.; Gulati, S.; Aggarwal, M.
Ano da publicação	2010
Conferência	Computer and Communication Technology (ICCCT), 2010 International Conference on

Resumo

O artigo descreve uma abordagem de predição de propensão a falhas usando uma evolução do algoritmo de clusterização *K-means*. Além das métricas de código-fonte, utiliza também métricas de requisitos.

QS1: De onde são obtidos os dados para a predição?

Bases de medidas: NASA MDP

QS2: Qual a granularidade dos artefatos de software analisados para construir a predição?

Não é especificado

QS3: Que tipos de características estruturais são usados?

Métricas de Código

QS4: Para que propósito a abordagem é destinada?

Predição de Falhas

QS5: Qual a técnica utilizada para construir a predição?

Categoria: Clusterização Nome: *K-Means-Sorensen* Descrição: O algoritmo de clusterização *K-means* é utilizado com outra função de distância, a distância de *Sorensen* ou distância de *Bray Curtis*. O algoritmo modificado (abordagem) é referido como algoritmo de clusterização *K-Means-Sorensen*.
QS6: Quais métricas de desempenho foram utilizadas para avaliar a abordagem?
Cobertura, Taxa de Falsos Positivos, Precisão e Acurácia.
QS7: Qual é o desempenho aproximado das predições fornecido pela abordagem?
Cobertura = 99%
QS8: Oferece apoio ferramental?
Não

Dados da Publicação	
Título	A Comparative Study Of Predictive Models For Program Changes During System Testing And Maintenance
Autores	Khoshgoftaar, T.M.; Munson, J.C.; Lanning, D.L.
Ano da publicação	1993
Conferência	Software Maintenance ,1993. CSM-93, Proceedings., Conference on

Resumo
O artigo propõe uma abordagem que melhora o desempenho das técnicas de predição que utilizam a técnica Multivariate Regression. Para tal, utilizam-se técnicas como Análise de Componentes Principais e Análise de Cluster para reduzir a dimensionalidade das variáveis independentes do modelo. Em seguida, o modelo de regressão é construído e, posteriormente, utilizado.
QS1: De onde são obtidos os dados para a predição?
Projeto de software: *Medical Imaging System*
QS2: Qual a granularidade dos artefatos de software analisados para construir a predição?
Módulo
QS3: Que tipos de características estruturais são usados?
Métricas de Código: Número de Linhas de Código (com e sem comentários), Caracteres, Comentários, Caracteres nos Comentários, Caracteres no Código; Complexidade Ciclomática, etc
QS4: Para que propósito a abordagem é destinada?
Predição de Modificações: Número de Mudanças
QS5: Qual a técnica utilizada para construir a predição?
Categoria: Regressão Nome: *Multivariate Regression*
QS6: Quais métricas de desempenho foram utilizadas para avaliar a abordagem?
Erro Absoluto
QS7: Qual é o desempenho aproximado das predições fornecido pela abordagem?
Média (Erro Absoluto) = 4,95
QS8: Oferece apoio ferramental?
Não.

Dados da Publicação	
Título	A Density Based Clustering Approach For Early Detection Of Fault Prone Modules
Autores	Sandhu, P.S.; Kaur, M.; Kaur, A.
Ano da publicação	2010
Conferência	Electronics and Information Engineering (ICEIE), 2010 International Conference On

Resumo
O artigo apresenta uma metodologia para a predição de propensão a falhas (*fault proneness*) que utiliza uma técnica de clusterização baseada em densidade (*Density-based Clustering*). São utilizadas métricas de código combinadas com métricas de requisitos.

QS1: De onde são obtidos os dados para a predição?
Bases de medidas: NASA MDP
QS2: Qual a granularidade dos artefatos de software analisados para construir a predição?
Módulo
QS3: Que tipos de características estruturais são usados?
Métricas de Código: Métricas de McCabe, de Halstead, de linhas de código, etc
QS4: Para que propósito a abordagem é destinada?
Predição de Falhas: *Fault-proneness*
QS5: Qual a técnica utilizada para construir a predição?
Categoria: Clusterização Nome: *Density-based cluster*
QS6: Quais métricas de desempenho foram utilizadas para avaliar a abordagem?
Cobertura, Taxa de Falsos Positivos, Precisão e Acurácia
QS7: Qual é o desempenho aproximado das predições fornecido pela abordagem?
Acurácia = 92.8%
QS8: Oferece apoio ferramental?
Não

Dados da Publicação	
Título	A Model To Predict Anti-Regressive Effort In Open Source Software
Autores	Capiluppi, A.; Fernandez-Ramil, J.
Ano da publicação	2007
Conferência	Software Maintenance, 2007. ICSM 2007. IEEE International Conference on
Resumo	
O artigo apresenta um modelo para predição de esforço antirregressivo (esforço de redução da complexidade do software). A partir de medidas estruturais e do Número de *Releases* com Mudanças, é construída uma lista em ordem decrescente com recomendações das funções às quais se deverá dar maior atenção durante o desenvolvimento da próxima *release*, com vistas a atingir o objetivo de diminuir sua complexidade.	
QS1: De onde são obtidos os dados para a predição?	
Projeto de software: projetos *open-source*	
QS2: Qual a granularidade dos artefatos de software analisados para construir a predição?	
Método	
QS3: Que tipos de características estruturais são usados?	
Métricas de Código: Número de Linhas de Código, Complexidade Ciclomática Métricas de Dependência: Acoplamento (contagem de chamadas de função)	
QS4: Para que propósito a abordagem é destinada?	
Predição de Modificações: recomendação de que funções são as melhores candidatas para se realizar o controle de complexidade (esforço antirregressivo)	
QS5: Qual a técnica utilizada para construir a predição?	
Categoria: Técnica Específica Nome: Não possui Descrição: As predições (ou são recomendações) são obtidas a partir de um ranqueamento das funções de acordo com suas complexidade (tamanho, complexidade ciclomática ou acoplamento) e medida do Número de *Releases* com Mudanças (isto é, o número de releases em que uma determinada função foi modificada). A aplicação de esforços antirregressivos em funções no topo da lista trará os maiores benefícios.	
QS6: Quais métricas de desempenho foram utilizadas para avaliar a abordagem?	
Avaliação Específica: Utilizou-se a medida de avaliação definida como Número de Acertos (*Hits*): das recomendações geradas, são obtidas as 'q' primeiras, sendo esse 'q' correspondente ao número de funções em que realmente houve modificações para uma determinada release	

em estudo (chame-se essa lista de “comportamento observado”). É contado, então, dessas q recomendações, quantas estão presente na lista de “comportamento observado”. A taxa de acerto será constituída por essa contagem dividida por 'q'.
QS7: Qual é o desempenho aproximado das predições fornecido pela abordagem?
Avaliação Específica: Taxa de Acertos = 70%
QS8: Oferece apoio ferramental?
Não

Dados da Publicação	
Título	A Neural Network Approach For Predicting Software Development Faults
Autores	Khoshgoftaar, T.M.; Pandya, A.S.; More, H.B.
Ano da publicação	1992
Conferência	Software Reliability Engineering, 1992. Proceedings., Third International Symposium on
Resumo	
O artigo propõe um preditor baseado em Redes Neurais e o compara a outro baseado em Regressão Linear Multivariada.	
QS1: De onde são obtidos os dados para a predição?	
Projeto de software: Ambiente de Desenvolvimento ADA para Controle e Comando de um Sistema Militar de Comunicações (CCCS)	
QS2: Qual a granularidade dos artefatos de software analisados para construir a predição?	
Módulo	
QS3: Que tipos de características estruturais são usados?	
Métrica de Código: Métricas de Halstead, Complexidade Ciclomática de McCabe, Número de Linhas de Còdigo, Número de Linhas em Branco, etc	
QS4: Para que propósito a abordagem é destinada?	
Predição de Qualidade	
QS5: Qual a técnica utilizada para construir a predição?	
Categoria: Redes Neurais	
QS6: Quais métricas de desempenho foram utilizadas para avaliar a abordagem?	
Erro Relativo	
QS7: Qual é o desempenho aproximado das predições fornecido pela abordagem?	
Média (Erro Relativo) = 39,8%	
QS8: Oferece apoio ferramental?	
Não	

Dados da Publicação	
Título	A Probabilistic Approach To Predict Changes In Object-Oriented Software Systems
Autores	Sharafat, Ali R.; Tahvildari, Ladan
Ano da publicação	2007
Conferência	Software Maintenance and Reengineering, 2007. CSMR '07. 11th European Conference on
Resumo	
O artigo apresenta uma abordagem probabilística para predição da ocorrência de modificações na próxima *release* de uma classe. Essa predição é baseada em probabilidades.	
QS1: De onde são obtidos os dados para a predição?	
Projeto de software Diagrama: Diagramas UML	
QS2: Qual a granularidade dos artefatos de software analisados para construir a predição?	
Método	
QS3: Que tipos de características estruturais são usados?	
Métricas de Código: Complexidade Ciclomática, Número de Linhas de Código, Número Máximo de Instruções de Desvio, Número de Variáveis Locais, Número de Parâmetros, Message Passing Coupling	

Métricas OO: Acesso a Dados Locais Métricas de Dependência: Acesso a Dados Importados, Número de Classes Importadas
QS4: Para que propósito a abordagem é destinada?
Predição de Modificações: probabilidade de ocorrência de modificações em uma classe na próxima release
QS5: Qual a técnica utilizada para construir a predição?
Categoria: Técnica específica Nome: Não possui Descrição: Para determinar que classes serão modificadas em um sistema na próxima release, a abordagem probabilística proposta usa o código-fonte do sistema. É utilizada uma fórmula obtida a partir do teorema de Bayes. Essa fórmula é composta pela Probabilidade de Mudança Interna (probabilidade de que uma classe será modificada devido a mudanças originadas na própria classe) e pela Probabilidade de Propagação (probabilidade de que uma mudança se propague de uma classe a outra). O resultado dessa fórmula é a probabilidade total de mudança para cada classe. Deve-se ressaltar que o cálculo da Probabilidade de Mudança Interna é baseado no valor de métricas estruturais.
QS6: Quais métricas de desempenho foram utilizadas para avaliar a abordagem?
Taxa de Falsos Positivos, Cobertura Outros: Taxa de Falsos Negativos
QS7: Qual é o desempenho aproximado das predições fornecido pela abordagem?
Cobertura = 50%
QS8: Oferece apoio ferramental?
Não

Dados da Publicação	
Título	A Two-Step Model For Defect Density Estimation
Autores	Kutlubay, O.; Turhan, B.; Bener, A.B.
Ano da publicação	2007
Conferência	Software Engineering and Advanced Applications, 2007. 33rd EUROMICRO Conference on

Resumo
O artigo apresenta uma abordagem de dois passos para predição de densidade de defeitos. No primeiro passo, módulos são classificados como defectivos e não defectivos. No segundo passo, os dados dos módulos classificados como defectivos são utilizados em uma análise de regressão. Como resultado, a abordagem fornece uma estimativa das densidades de defeitos em módulos defectivos.
QS1: De onde são obtidos os dados para a predição?
Bases de medidas: NASA MDP
QS2: Qual a granularidade dos artefatos de software analisados para construir a predição?
Módulo
QS3: Que tipos de características estruturais são usados?
Métricas de Código
QS4: Para que propósito a abordagem é destinada?
Predição de Falhas: densidade de defeitos em módulos defectivos
QS5: Qual a técnica utilizada para construir a predição?
Categoria: Híbrido de Classificação e Regressão Descrição: Utilização de Naive Bayes e Decision Tree na fase de classificação e Radial Basis Function e Regression Tree na fase de regressão.
QS6: Quais métricas de desempenho foram utilizadas para avaliar a abordagem?
Acurácia, Taxa de Falsos Positivos, Cobertura Outros: Raiz Quadrada da Média da Soma Residual de Quadrados (RMSE)
QS7: Qual é o desempenho aproximado das predições fornecido pela abordagem?
Outros: RMSE = 26,12
QS8: Oferece apoio ferramental?
Não

Dados da Publicação	
Título	Adding Process Metrics To Enhance Modification Complexity Prediction
Autores	Toth, G.; Vegh, A.Z.; Beszedes, A.; Gyimothy, T.
Ano da publicação	2011
Conferência	Program Comprehension (ICPC), 2011 IEEE 19th International Conference on

Resumo
O artigo mostra os resultados da utilização de um *framework* (não detalhado nem referenciado no texto) aplicado para o propósito da predição da complexidade da modificação. A abordagem proposta utiliza uma combinação de métricas de produto e de processo em algoritmos de classificação tradicionais.
QS1: De onde são obtidos os dados para a predição?
Sistema de Gerência de Configuração: Subversion
QS2: Qual a granularidade dos artefatos de software analisados para construir a predição?
Classe
QS3: Que tipos de características estruturais são usados?
Métricas de Código: Número de Linhas de Código Métricas OO: Depth of Inheritance Tree, Número de Filhos Métricas de Dependência: Acoplamento entre Classes
QS4: Para que propósito a abordagem é destinada?
Predição de Modificações: Complexidade de Modificação
QS5: Qual a técnica utilizada para construir a predição?
Categoria: Classificação Descrição: Utilização do framework com as técnicas J48, RBF e ClassificationViaRegression.
QS6: Quais métricas de desempenho foram utilizadas para avaliar a abordagem?
Precisão, Cobertura
QS7: Qual é o desempenho aproximado das predições fornecido pela abordagem?
Precisão = 43,3%
QS8: Oferece apoio ferramental?
Sim

Dados da Publicação	
Título	An Empirical Approach For Software Fault Prediction
Autores	Kaur, A.; Brar, A.S.; Sandhu, P.S.
Ano da publicação	2010
Conferência	Industrial and Information Systems (ICIIS), 2010 International Conference on

Resumo
O artigo propõe uma abordagem para predição de propensão a falhas (*fault-proneness*) que utiliza um algoritmo de clusterização. São utilizadas métricas de código em combinação com métricas de requisitos. Tal combinação é realizada através de uma operação de *inner join* entre os atributos de bases do repositório MDP.
QS1: De onde são obtidos os dados para a predição?
Bases de medidas: NASA MDP
QS2: Qual a granularidade dos artefatos de software analisados para construir a predição?
Módulo
QS3: Que tipos de características estruturais são usados?
Métricas de Código
QS4: Para que propósito a abordagem é destinada?
Predição de Falhas: *Fault-proneness*
QS5: Qual a técnica utilizada para construir a predição?

Categoria: Clusterização Nome: *Fuzzy C Means Clustering*
QS6: Quais métricas de desempenho foram utilizadas para avaliar a abordagem?
Cobertura, Taxa de Falsos Positivos
QS7: Qual é o desempenho aproximado das predições fornecido pela abordagem?
Cobertura = 100%
QS8: Oferece apoio ferramental?
Não

Dados da Publicação	
Título	Aode For Source Code Metrics For Improved Software Maintainability
Autores	Yingjie T., Chuanliang C., Chunhua Z.
Ano da publicação	2008
Conferência	Proceedings of the 4th International Conference on Semantics, Knowledge, and Grid, SKG 2008
Resumo	
O artigo apresenta um novo método de classificação, chamado AODE, para a predição da quantidade de defeitos (dada em categorias: zero defeitos, 1 a 10 defeitos e mais de 10 defeitos). Antes de executar o AODE, é realizada uma seleção de atributos relevantes através de um método, também novo, chamado *Symmetrical Uncertainty*.	
QS1: De onde são obtidos os dados para a predição?	
Projeto de software: Sistema General Electric's SIGNA	
QS2: Qual a granularidade dos artefatos de software analisados para construir a predição?	
Módulo	
QS3: Que tipos de características estruturais são usados?	
Métricas de Código: Número de Linhas de Código, Número de Linhas Executáveis, Número Total de Caracteres, Número de Linhas de Comentários, Número de Halstead, Complexidade Ciclomática de McCabe, etc	
QS4: Para que propósito a abordagem é destinada?	
Predição de Falhas: Quantidade categórica de defeitos	
QS5: Qual a técnica utilizada para construir a predição?	
Categoria: Classificação Nome: *Aggregating One-Dependency Estimators*	
QS6: Quais métricas de desempenho foram utilizadas para avaliar a abordagem?	
Outros: Média Ponderada das Precisões, Média Ponderada das Coberturas	
QS7: Qual é o desempenho aproximado das predições fornecido pela abordagem?	
Outros: Máximo (Média Ponderada (Precisão)) = 79,33%	
QS8: Oferece apoio ferramental?	
Não	

Dados da Publicação	
Título	Application Of Neural Network For Predicting Software Development Faults Using Object-Oriented Design Metrics
Autores	Mie Mie Thet Thwin; Tong-Seng Quah
Ano da publicação	2002
Conferência	Neural Information Processing, 2002. ICONIP '02. Proceedings of the 9th International Conference on
Resumo	
O artigo apresenta uma abordagem para predição do número de falhas através do uso de Redes Neurais. Mais especificamente, a técnica utilizada foi a Rede de Warp (uma rede neural do tipo *Backpropagation*). As variáveis de entrada para a rede neural são métricas orientadas a objeto.	
QS1: De onde são obtidos os dados para a predição?	

Projeto de software: Subsistemas de um sistema de Interface Homem-Máquina
QS2: Qual a granularidade dos artefatos de software analisados para construir a predição?
Classe
QS3: Que tipos de características estruturais são usados?
Métricas OO: Depth of Inheritance Tree, Número de Filhos, Weighted Method Count, Complexidade Média dos métodos Métricas de Dependência: Acoplamento entre Objetos, Acoplamento de Herança, Acoplamento entre Métodos, Response for a Class
QS4: Para que propósito a abordagem é destinada?
Predição de Falhas: Número de falhas
QS5: Qual a técnica utilizada para construir a predição?
Categoria: Redes Neurais Nome: Rede de Warp
QS6: Quais métricas de desempenho foram utilizadas para avaliar a abordagem?
Outros: Coeficiente de múltipla determinação, Coeficiente de correlação, Média da Soma Residual de Quadrados Erro Absoluto
QS7: Qual é o desempenho aproximado das predições fornecido pela abordagem?
Média (Erro Absoluto) = 0,787345
QS8: Oferece apoio ferramental?
Não

Dados da Publicação	
Título	Comparison Of Conventional Approaches And Soft-Computing Approaches For Software Quality Prediction
Autores	Baisch, E.; Liedtke, T.
Ano da publicação	1997
Conferência	Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on

Resumo
O artigo apresenta uma abordagem que utiliza um sistema *Fuzzy* para classificação. Esse sistema é gerado com a ajuda de um algoritmo genético.
QS1: De onde são obtidos os dados para a predição?
Projeto de software: Sistema Alcatel 1000 S12
QS2: Qual a granularidade dos artefatos de software analisados para construir a predição?
Módulo
QS3: Que tipos de características estruturais são usados?
Métricas de Código: Número de comandos executáveis, Complexidade do comando, Complexidade da Expressão, Complexidade dos dados, Profundidade dos aninhamentos, etc.
QS4: Para que propósito a abordagem é destinada?
Predição de Modificações
QS5: Qual a técnica utilizada para construir a predição?
Categoria: Classificação Nome: *Fuzzy Expert-System* Descrição: *Fuzzy Expert-System* gerado por algoritmos genético
QS6: Quais métricas de desempenho foram utilizadas para avaliar a abordagem?
Outros: Número de Verdadeiros Positivos, Número de Falsos Positivos, Número de Verdadeiros Negativos, Número de Falsos Negativos
QS7: Qual é o desempenho aproximado das predições fornecido pela abordagem?
Outros: Número de Verdadeiros Positivos = 84%
QS8: Oferece apoio ferramental?
Não

Dados da Publicação	
Título	Design-Level Metrics Estimation Based On Code Metrics
Autores	Sami A., Fakhrahmad S.M.
Ano da publicação	2010
Conferência	Proceedings of the ACM Symposium on Applied Computing

Resumo

O uso de métricas de código e métricas em nível de projeto (extraídas de diagramas) torna a predição através de classificadores mais acurada. No entanto, obter medidas em nivel de projeto é desafiador e nem sempre é uma tarefa simples. Para contornar essa dificuldade, o artigo propõe um sistema de predição de defeitos que estima o valor de métricas em nível de projeto e dependências com base em métricas de código. Dessa forma, a quantidade de métricas a serem incluídas na entrada do sistema, bem como a dificuldade de obtê-las, é reduzida. O sistema é dividido em três partes: (1) Minerador de Dependências, que visa encontrar combinações entre métricas de código que descrevam métricas em nível de projeto; (2) Subsistema *Fuzzy* de Construção de Estimativa, que utiliza sistemas de modelagem *fuzzy* para estimar o valor da métrica em nível de projeto a partir das métricas de código relacionadas (encontradas no passo 1); (3) Subsistema de Classificação *Fuzzy* baseado em Regras (*Rule-based*), que realiza a predição da propensão a falhas de determinado módulo.

QS1: De onde são obtidos os dados para a predição?

Bases de medidas: NASA MDP

QS2: Qual a granularidade dos artefatos de software analisados para construir a predição?

Módulo

QS3: Que tipos de características estruturais são usados?

Métricas de Código: Número de comandos de decisão, Número de operandos, Número de Linhas em branco, Número de Linhas de Comentários, Número de parâmetros, etc

QS4: Para que propósito a abordagem é destinada?

Predição de Defeitos/Falhas: *Fault-proneness*

QS5: Qual a técnica utilizada para construir a predição?

Categoria: Classificação

Nome: *Fuzzy Rule-based Classification System*

QS6: Quais métricas de desempenho foram utilizadas para avaliar a abordagem?

Acurácia

QS7: Qual é o desempenho aproximado das predições fornecido pela abordagem?

Acurácia = 85%

QS8: Oferece apoio ferramental?

Sim

Dados da Publicação	
Título	Early Software Fault Prediction Using Real Time Defect Data
Autores	Kaur, A.; Sandhu, P.S.; Bra, A.S.
Ano da publicação	2009
Conferência	Machine Vision, 2009. ICMV '09. Second International Conference on

Resumo

O artigo apresenta uma abordagem para predição de falhas através de um algoritmo de clusterização.

QS1: De onde são obtidos os dados para a predição?

Bases de medidas: NASA MDP

QS2: Qual a granularidade dos artefatos de software analisados para construir a predição?

Módulo

QS3: Que tipos de características estruturais são usados?

Métricas de Código

QS4: Para que propósito a abordagem é destinada?

Predição de Falhas
QS5: Qual a técnica utilizada para construir a predição?
Categoria: Clusterização Nome: *K-Means*
QS6: Quais métricas de desempenho foram utilizadas para avaliar a abordagem?
Cobertura, Taxa de Falsos Positivos
QS7: Qual é o desempenho aproximado das predições fornecido pela abordagem?
Cobertura = 100%
QS8: Oferece apoio ferramental?
Não

Dados da Publicação	
Título	Evaluating Architectural Stability Using A Metric-Based Approach
Autores	Tonu S.A., Ashkan A., Tahvildari L.
Ano da publicação	2006
Conferência	Proceedings of the European Conference on Software Maintenance and Reengineering, CSMR
Resumo	
O artigo apresenta um framework baseado em métricas para realizar a avaliação da estabilidade de sistemas. A abordagem é conduzida em 3 etapas: 1- Extração da arquitetura do sistema-alvo a partir da análise de seu código-fonte; 2- Análise retrospectiva da evolução da estabilidade do sistema ao longo do tempo; 3- Determinação (predição) de um conjunto de mudanças prováveis e de seu impacto ao serem aplicadas.	
QS1: De onde são obtidos os dados para a predição?	
Projeto de software: Códigos-fontes de *releases* sucessivas de um projeto	
QS2: Qual a granularidade dos artefatos de software analisados para construir a predição?	
Módulo, Subsistema	
QS3: Que tipos de características estruturais são usados?	
Métricas de Código: Número de Linhas de Código, Número de Funções, Número de Chamadas de Função (em/entre subsistemas)	
QS4: Para que propósito a abordagem é destinada?	
Outros: Avaliação da estabilidade arquitetural do código-fonte (incluindo a Predição de Mudanças e de seu impacto)	
QS5: Qual a técnica utilizada para construir a predição?	
Categoria: Não é especificado	
QS6: Quais métricas de desempenho foram utilizadas para avaliar a abordagem?	
Não houve validação experimental	
QS7: Qual é o desempenho aproximado das predições fornecido pela abordagem?	
Não houve validação experimental	
QS8: Oferece apoio ferramental?	
Sim	

Dados da Publicação	
Título	Evaluating Web Applications Testability By Combining Metrics And Analogies
Autores	Marchetto, A.; Trentini, A.
Ano da publicação	2005
Conferência	Information and Communications Technology, 2005. Enabling Technologies for the New Knowledge Society: ITI 3rd International Conference on
Resumo	
O artigo descreve um modelo baseado em métricas para ajudar o usuário a medir um sistema de software Web e analisar/predizer fatores como testabilidade, confiabilidade, propensão a erros e tolerância a falhas. A predição é realizada através do algoritmo de classificação *k-Nearest Neighbour*.	

QS1: De onde são obtidos os dados para a predição?
Projeto de software: Aplicações Web de código aberto
QS2: Qual a granularidade dos artefatos de software analisados para construir a predição?
Módulo
QS3: Que tipos de características estruturais são usados?
Métricas de Código: Número de Linhas de Código, Número de Módulo, Weighted Method Count, Número de Atributos do Módulo Métricas de Dependência: Número de Dependências Cíclicas, Acoplamento entre Componentes
QS4: Para que propósito a abordagem é destinada?
Predição de Qualidade: Testabilidade, Confiabilidade Predição de Defeitos/Falhas: Propensão a Erros, Tolerância a Falhas
QS5: Qual a técnica utilizada para construir a predição?
Categoria: Classificação Nome: *K-Nearest Neighbour*
QS6: Quais métricas de desempenho foram utilizadas para avaliar a abordagem?
Erro Relativo Avaliação específica: PRED[X]: percentual dos elementos com erro relativo menor ou igual a X, em relação ao número total de instâncias do experimento
QS7: Qual é o desempenho aproximado das predições fornecido pela abordagem?
Avaliação específica: PRED[39] = 50%
QS8: Oferece apoio ferramental?
Não

Dados da Publicação	
Título	Improving Code Churn Predictions During The System Test And Maintenance Phases
Autores	Khoshgoftaar, T.M.; Szabo, R.M.
Ano da publicação	1994
Conferência	Software Maintenance, 1994. Proceedings., International Conference on
Resumo	
O artigo propõe o uso de Redes Neurais para melhorar o desempenho da predição de modificações.	
QS1: De onde são obtidos os dados para a predição?	
Projeto de software: Sistemas de interface entre o *kernel* e seu hardware	
QS2: Qual a granularidade dos artefatos de software analisados para construir a predição?	
Módulo	
QS3: Que tipos de características estruturais são usados?	
Métricas de Código: Número de Operadores/Operandos Únicos, Número Total de Operadores/Operandos, Número de Comandos Executáveis, Complexidade Ciclomática de McCabe, Métricas de Dependência: Fan-In, Fan-Out, etc	
QS4: Para que propósito a abordagem é destinada?	
Predição de Modificações	
QS5: Qual a técnica utilizada para construir a predição?	
Categoria: Redes Neurais	
QS6: Quais métricas de desempenho foram utilizadas para avaliar a abordagem?	
Erro Absoluto	
QS7: Qual é o desempenho aproximado das predições fornecido pela abordagem?	
Média (Erro Absoluto) = 1965	
QS8: Oferece apoio ferramental?	
Não	

Dados da Publicação	
Título	Tempest: Towards Early Identification Of Failure-Prone Binaries
Autores	Bhat, T.; Nagappan, N.
Ano da publicação	2008
Conferência	Dependable Systems and Networks With FTCS and DCC, 2008. DSN 2008. IEEE International Conference on
Resumo	
O artigo apresenta uma ferramenta, *Tempest*, cuja finalidade é predizer a propensão a falhas de arquivos binários e prover uma análise das métricas de complexidade de código baseada em dados históricos.	
QS1: De onde são obtidos os dados para a predição?	
Arquivos Binários	
QS2: Qual a granularidade dos artefatos de software analisados para construir a predição?	
Módulo	
QS3: Que tipos de características estruturais são usados?	
Métricas de Código: Complexidade Ciclomática, Número de Linhas de Código, Número de Variáveis Globais, Número Total de Funções, Número de Linhas, Número de Parâmetros Métricas de Dependência: Fan-In, Fan-Out	
QS4: Para que propósito a abordagem é destinada?	
Predição de Defeitos/Falhas: Fault-proneness	
QS5: Qual a técnica utilizada para construir a predição?	
Categoria: Regressão Nome: Regressão Estatística	
QS6: Quais métricas de desempenho foram utilizadas para avaliar a abordagem?	
Não houve validação experimental	
QS7: Qual é o desempenho aproximado das predições fornecido pela abordagem?	
Não houve validação experimental	
QS8: Oferece apoio ferramental?	
Sim	

Dados da Publicação	
Título	Software Testing: A Machine Learning Experiment
Autores	Cheatham Thomas J., Yoo Jungsoon P., Wahl Nancy J.
Ano da publicação	1995
Conferência	Proceedings - ACM Computer Science Conference
Resumo	
O artigo apresenta uma abordagem para a predição do tempo de um teste de software. Essa abordagem utiliza para predição o algoritmo de clusterização *COBWEB/3*, que é uma evolução de um algoritmo existente, o *COBWEB*.	
QS1: De onde são obtidos os dados para a predição?	
Não é especificado	
QS2: Qual a granularidade dos artefatos de software analisados para construir a predição?	
Não é especificado	
QS3: Que tipos de características estruturais são usados?	
Métricas de Código: Número de Linhas de Código Não Comentadas, Número de Funções, Nível de Aninhamento, Número de Instruções de Decisão, etc	
QS4: Para que propósito a abordagem é destinada?	
Predição de Tempo: Tempo de teste	
QS5: Qual a técnica utilizada para construir a predição?	
Categoria: Clusterização	

Nome: *COBWEB/3*
QS6: Quais métricas de desempenho foram utilizadas para avaliar a abordagem?
Não houve validação experimental
QS7: Qual é o desempenho aproximado das predições fornecido pela abordagem?
Não houve validação experimental
QS8: Oferece apoio ferramental?
Não

Dados da Publicação	
Título	Validation Of Network Measures As Indicators Of Defective Modules In Software Systems
Autores	Tosun A., Turhan B., Bener A.
Ano da publicação	2009
Conferência	ACM International Conference Proceeding Series
Resumo	
O artigo apresenta uma abordagem para a predição de defeitos que utiliza o algoritmo *Naive Bayes*. Para melhorar sua performance, foi incorporado o *Call Graph Based Ranking Framework*, um framework que gera para cada módulo um valor (*rank*) que é multiplicado por todas as métricas da entrada para ajustar seus valores, de modo que reflitam aspectos das interações intermodulares. Além de apresentar uma nova abordagem, o artigo replica os experimentos realizados em outra pesquisa, utilizando fontes de dados diferentes daquelas analisadas na pesquisa anterior.	
QS1: De onde são obtidos os dados para a predição?	
Projeto de software: *Releases* do Eclipse Bases de medidas: NASA PROMISE	
QS2: Qual a granularidade dos artefatos de software analisados para construir a predição?	
Módulo	
QS3: Que tipos de características estruturais são usados?	
Métricas de Código: Número Total de Operadores/Operandos, Número de Linhas de Comentários, Número de Linhas de Código, métricas de Halstead, etc	
QS4: Para que propósito a abordagem é destinada?	
Predição de Falhas	
QS5: Qual a técnica utilizada para construir a predição?	
Categoria: Classificação Nome: *Naive Bayes* Descrição: *Naive Bayes* usando *Call Graph Based Ranking Framework*	
QS6: Quais métricas de desempenho foram utilizadas para avaliar a abordagem?	
Cobertura, Precisão, Taxa de Falsos Positivos	
QS7: Qual é o desempenho aproximado das predições fornecido pela abordagem?	
Precisão = 72%	
QS8: Oferece apoio ferramental?	
Não	

Dados da Publicação	
Título	Utilizing Computational Intelligence In Estimating Software Readiness
Autores	Tong Seng Quah; Mie Mie Thet Thwin
Ano da publicação	2006
Conferência	Neural Networks, 2006. IJCNN '06. International Joint Conference on
Resumo	
O artigo mostra uma abordagem para predição do Número de Falhas (Faults), do Número de Linhas Modificadas por classe e do Tempo Necessário para realizar modificações, usando uma Rede Neural combinada com uma estratégia de treinamento baseada nos algoritmos	

genéticos.
QS1: De onde são obtidos os dados para a predição?
Projeto de software: Aplicações Gerenciadoras de *Warehouse* e Sistema de Informações Hwafuh
QS2: Qual a granularidade dos artefatos de software analisados para construir a predição?
Classe
QS3: Que tipos de características estruturais são usados?
Métricas OO: Número de Atributos, Complexidade Ciclomática Média/Total Métricas de Dependência: Acoplamento entre Objetos, Lack of Cohesion in Methods
QS4: Para que propósito a abordagem é destinada?
Predição de Defeitos/Falhas: Número de Falhas (Faults) Predição de Modificações: Número de Linhas Modificadas por classe Predição de Tempo: Tempo Necessário para realizar modificações
QS5: Qual a técnica utilizada para construir a predição?
Categoria: Redes Neurais Descrição: Redes Neurais combinadas com uma estratégia de treinamento baseada nos algoritmos genéticos
QS6: Quais métricas de desempenho foram utilizadas para avaliar a abordagem?
Outros: Coeficiente de Múltipla Determinação (R^2), Coeficiente de Correlação, Média da Soma de Erros Residuais, Raiz Quadrada da Média da Soma de Erros Residuais
QS7: Qual é o desempenho aproximado das predições fornecido pela abordagem?
Outros: $R^2 = 0.65$
QS8: Oferece apoio ferramental?
Não

Dados da Publicação	
Título	Using Statistical Models To Predict Software Regressions
Autores	Tarvo, A.
Ano da publicação	2008
Conferência	Software Reliability Engineering, 2008. ISSRE 2008. 19th International Symposium on
Resumo	
O artigo mostra um sistema para predição de erros de regressão (i.e., mudanças indesejadas no comportamento de funcionalidades já estabilizadas do software), o qual utiliza a técnica *Stepwise Logistic Regression* e tem como entradas métricas de código, de dependências e de modificações.	
QS1: De onde são obtidos os dados para a predição?	
Projeto de software: Windows XP	
QS2: Qual a granularidade dos artefatos de software analisados para construir a predição?	
Módulo, Método	
QS3: Que tipos de características estruturais são usados?	
Métricas de Código: Número de Parâmetros Globais, Tamanho do Componente Métricas OO: Número de Classes no módulo, Número de Métodos na Classe Métricas de Dependência: Número de Dependências Externas e Número de Dependências Internas	
QS4: Para que propósito a abordagem é destinada?	
Predição de Risco: Risco de erros de regressão	
QS5: Qual a técnica utilizada para construir a predição?	
Categoria: Regressão Nome: *Stepwise Logistic Regression*	
QS6: Quais métricas de desempenho foram utilizadas para avaliar a abordagem?	
Precisão, Cobertura, Taxa de Falsos Positivos Outros: Valor Médio (μ) da Área Abaixo da Curva ROC	

QS7: Qual é o desempenho aproximado das predições fornecido pela abordagem?
Outros: $\mu = 0,77$
QS8: Oferece apoio ferramental?
Sim

Dados da Publicação	
Título	Squaner: A Framework For Monitoring The Quality Of Software Systems
Autores	Haderer, N.; Khomh, F.; Antoniol, G.
Ano da publicação	2010
Conferência	Software Maintenance (ICSM), 2010 IEEE International Conference on
Resumo	
O artigo apresenta o framework *Squaner*. A cada commit feito por um desenvolvedor, o *Squaner* se conecta diretamente ao repositório SVN de um projeto, extrai o código-fonte, realiza detecção de padrões de projeto, antipadrões e anomalias, realiza avaliações sobre a qualidade do software e predições de falhas (*faults*).	
QS1: De onde são obtidos os dados para a predição?	
Sistema de Gerência de Configuração: Subversion	
QS2: Qual a granularidade dos artefatos de software analisados para construir a predição?	
Classe	
QS3: Que tipos de características estruturais são usados?	
Métricas de Código: Complexidade Ciclomática Média/Total Métricas de Dependência: Acoplamento entre Objetos, Lack of Cohesion in Methods, etc	
QS4: Para que propósito a abordagem é destinada?	
Predição de Falhas: Probabilidade de ocorrência de falhas nos próximos 6 meses	
QS5: Qual a técnica utilizada para construir a predição?	
Categoria: Outros Nome: *Bayesian Belief Network*	
QS6: Quais métricas de desempenho foram utilizadas para avaliar a abordagem?	
Não houve validação experimental	
QS7: Qual é o desempenho aproximado das predições fornecido pela abordagem?	
Não houve validação experimental	
QS8: Oferece apoio ferramental?	
Sim	

Dados da Publicação	
Título	Statistically-Based Program Size Estimation
Autores	Takahashi, M.; Miyake, T.; Hanata, S.
Ano da publicação	1989
Conferência	Computer Software and Applications Conference, 1989. COMPSAC 89., Proceedings of the 13th Annual International
Resumo	
O artigo apresenta uma fórmula para a predição do tamanho do programa. Essa fórmula é formada por métricas de Halstead extraídas de diagramas de lógica do programa escritos em notação HCP. Por utilizar os diagramas HCP como fonte de dados, a abordagem permite que se conheça o tamanho que terá o programa antes da fase de implementação.	
QS1: De onde são obtidos os dados para a predição?	
Diagrama: Diagramas HCP	
QS2: Qual a granularidade dos artefatos de software analisados para construir a predição?	
Módulo	
QS3: Que tipos de características estruturais são usados?	

Métricas de Código: Número de Operadores (n_1) e Operandos (n_2) distintos, Número Total de Operadores (N_1) e Operandos (N_2), Tamanho do Programa ($N = N_1 + N_2$) e Volume do Programa ($V = N * \log_2 (n_1 + n_2)$)
QS4: Para que propósito a abordagem é destinada?
Predição de Métricas de Código: Tamanho do programa
QS5: Qual a técnica utilizada para construir a predição?
Categoria: Técnica Específica Nome: Não possui Descrição: É utilizado o modelo: $s = a_0 + a_1 * f_1 + \sum_{j=1...2} (a_{2j} \times f_{2j})$ Onde: s : tamanho do programa (KLOC), f_1 : complexidade do software (medido com N_1) f_{2j} : a j-ésima categoria de habilidade do programador, dentre duas possíveis (mais de 6 anos de experiência em programação e menos de 5 anos de experiência em programação) a_0 ,a_1, a_{2j} (j = 1,2,...) : parâmetros, definidos através da técnica de Análise de Regressão Múltipla.
QS6: Quais métricas de desempenho foram utilizadas para avaliar a abordagem?
Erro Relativo
QS7: Qual é o desempenho aproximado das predições fornecido pela abordagem?
Erro Relativo = 20%
QS8: Oferece apoio ferramental?
Não

Dados da Publicação	
Título	Software Science Model Of Compile Time
Autores	Shaw Jr. Wade H., Howatt James W., Maness Robert S., Miller Dennis M.
Ano da publicação	1992
Conferência	IEEE Transactions on Software Engineering

Resumo
O artigo apresenta uma abordagem para a predição do tempo de compilação para compiladores Ada. O modelo de predição é uma fórmula composta por métricas de Halstead e é baseado no trabalho de ciência do software (*software science metrics*) desse mesmo autor.
QS1: De onde são obtidos os dados para a predição?
Projeto de software: Prototype Ada Compiler Evaluation Capability
QS2: Qual a granularidade dos artefatos de software analisados para construir a predição?
Módulo
QS3: Que tipos de características estruturais são usados?
Métricas de Código: Número de Operadores (n_1) e Operandos (n_2) distintos, Número Total de Operadores (N_1) e Operandos (N_2), Tamanho do Programa ($N = N_1 + N_2$), Volume do Programa ($V = N * \log_2 (n_1 + n_2)$) e Volume Potencial do Programa ($V^* = [2 + n^*_2] * [\log_2(2 + n^*_2)]$) OBS: n^*_2 é o número de parâmetros de entrada/saída
QS4: Para que propósito a abordagem é destinada?
Predição de Tempo: Tempo de compilação
QS5: Qual a técnica utilizada para construir a predição?
Categoria: Técnica específica Nome: Não possui Descrição: É utilizado o seguinte modelo: $T = KV^a(V^*)^b$ onde: K : taxa ou velocidade de compilação V : volume do programa

V* : volume potencial do programa Para a execução dos experimentos, o modelo é linearizado para: ln (T) = ln (K) + a * ln (V) + b * ln (V *) + error
QS6: Quais métricas de desempenho foram utilizadas para avaliar a abordagem?
Outros: Coeficiente de Múltipla Determinação (R^2)
QS7: Qual é o desempenho aproximado das predições fornecido pela abordagem?
Outros: $R^2 = 0,7179$
QS8: Oferece apoio ferramental?
Não

Dados da Publicação	
Título	Software Quality Prediction Using Mixture Models With Em Algorithm
Autores	Ping Guo; Lyu, M.R.
Ano da publicação	2000
Conferência	Quality Software, 2000. Proceedings. First Asia-Pacific Conference on
Resumo	
O artigo mostra uma abordagem para a predição de propensão a falhas (*fault-proneness*) que utiliza o algoritmo *Mixture Model with Expectation-Maximum*.	
QS1: De onde são obtidos os dados para a predição?	
Projeto de software: Sistema Médico de Informações	
QS2: Qual a granularidade dos artefatos de software analisados para construir a predição?	
Módulo	
QS3: Que tipos de características estruturais são usados?	
Métricas de Código: Número Total de Linhas de Código, Número de Linhas de Código, Número de Caracteres, Número de Comentários, Métricas de Halstead e de Jansen, Complexidade Ciclomática de McCabe, etc	
QS4: Para que propósito a abordagem é destinada?	
Predição de Defeitos/Falhas: *Fault-proneness*	
QS5: Qual a técnica utilizada para construir a predição?	
Categoria: Classificação Nome: *Mixture Model with Expectation-Maximum*	
QS6: Quais métricas de desempenho foram utilizadas para avaliar a abordagem?	
Não é especificado	
QS7: Qual é o desempenho aproximado das predições fornecido pela abordagem?	
Não é especificado	
QS8: Oferece apoio ferramental?	
Não	

Dados da Publicação	
Título	Software Change Classification Using Hunk Metrics
Autores	Ferzund J., Ahsan S.N., Wotawa F.
Ano da publicação	2009
Conferência	IEEE International Conference on Software Maintenance, ICSM
Resumo	
O artigo apresenta uma abordagem para a predição de defeitos. Essa abordagem utiliza para a predição o algoritmo de classificação *Random Forest*, utilizando métricas extraídas de deltas (*hunks*) de revisões do Sistema de Gerência de Configuração.	
QS1: De onde são obtidos os dados para a predição?	
Sistema de Gerência de Configuração	

QS2: Qual a granularidade dos artefatos de software analisados para construir a predição?
Delta (*hunks*) de revisões do SGC
QS3: Que tipos de características estruturais são usados?
Métricas de Código: Número de Expressões Condicionais (presentes em um delta), Número de Loops (em um delta), Número de Atribuições (em um delta), Número de Operadores Relacionais (em um delta), Número de Breaks (em um delta), Número de Classes (em um delta), etc
QS4: Para que propósito a abordagem é destinada?
Predição de Falhas
QS5: Qual a técnica utilizada para construir a predição?
Categoria: Classificação Nome: *Random Forest*
QS6: Quais métricas de desempenho foram utilizadas para avaliar a abordagem?
Acurácia, Precisão, Cobertura, Outros
QS7: Qual é o desempenho aproximado das predições fornecido pela abordagem?
Acurácia = 81%
QS8: Oferece apoio ferramental?
Não

Dados da Publicação	
Título	Release Date Prediction For Telecommunication Software Using Bayesian Belief Networks
Autores	Ying Wang; Smith, M.
Ano da publicação	2002
Conferência	Electrical and Computer Engineering, 2002. IEEE CCECE 2002. Canadian Conference on
Resumo	
Neste artigo, dados de 5 produtos de software foram analisados com a intenção de se usar a informação obtida para predizer a data de apropriada para a *release* de uma nova versão de produtos similares, de modo que possuam qualidade equivalente às de suas versões anteriores. A predição é realizada através do algoritmo *Bayesian Belief Network*.	
QS1: De onde são obtidos os dados para a predição?	
Projeto de software: Softwares de uma companhia de telecomunicações	
QS2: Qual a granularidade dos artefatos de software analisados para construir a predição?	
Classe	
QS3: Que tipos de características estruturais são usados?	
Métricas de Código: Número de Linhas de Código, Número de Métodos, Complexidade Ciclomática Total Métricas de Dependência: Acoplamento entre Objetos, etc	
QS4: Para que propósito a abordagem é destinada?	
Predição de Tempo: Data de release	
QS5: Qual a técnica utilizada para construir a predição?	
Categoria: Outros Nome: *Bayesian Belief Network*	
QS6: Quais métricas de desempenho foram utilizadas para avaliar a abordagem?	
Não é especificado	
QS7: Qual é o desempenho aproximado das predições fornecido pela abordagem?	
Não é especificado	
QS8: Oferece apoio ferramental?	
Não	

Dados da Publicação	
Título	Does Calling Structure Information Improve The Accuracy Of Fault Prediction?

Autores	Shin Y., Bell R., Ostrand T., Weyuker E.
Ano da publicação	2009
Conferência	Proceedings of the 2009 6th IEEE International Working Conference on Mining Software Repositories, MSR 2009
Resumo	
O artigo apresenta a vantagem de utilizar atributos de *Calling Structure* como insumo para a predição de falhas. Propõe-se como modelo de predição é uma metodologia que utiliza a técnica de regressão *Negative Binomial Regression*.	
QS1: De onde são obtidos os dados para a predição?	
Projeto de software: Sistema de Manutenção de Negócios	
QS2: Qual a granularidade dos artefatos de software analisados para construir a predição?	
Módulo	
QS3: Que tipos de características estruturais são usados?	
Métricas de Código: Número de Linhas de Código Específica: Medidas das *Calling Structures*	
QS4: Para que propósito a abordagem é destinada?	
Predição de Falhas	
QS5: Qual a técnica utilizada para construir a predição?	
Categoria: Técnica Específica Descrição: A técnica *Negative Binomial Regression* é utilizada para calcular o número estimado de falhas das classes. As classes são ordenadas em ordem decrescente do tamanho estimado para cada uma delas e a porcentagem de falhas corretamente preditas é calculada para as primeiras 20% da lista ordenada. Para predizer falhas na Release N, constroem-se modelos usando os dados obtidos da Release 1 até a Release N-1. Então, aplica-se o modelo aos dados coletados para a Release N.	
QS6: Quais métricas de desempenho foram utilizadas para avaliar a abordagem?	
Avaliação específica: As estimativas foram ordenadas em ordem decrescente e foi calculada a Acurácia dos elementos do topo dessa lista (20% do total).	
QS7: Qual é o desempenho aproximado das predições fornecido pela abordagem?	
Avaliação específica: Acurácia = 78,29%	
QS8: Oferece apoio ferramental?	
Não	

Dados da Publicação	
Título	Improving Neural Network Predictions Of Software Quality Using Principal Components Analysis
Autores	Khoshgoftaar Taghi M., Szabo Robert M.
Ano da publicação	1994
Conferência	IEEE International Conference on Neural Networks - Conference Proceedings
Resumo	
O artigo explora a aplicação da Análise de Componentes Principais à modelagem de Redes Neurais com vistas a melhorar sua capacidade preditiva.	
QS1: De onde são obtidos os dados para a predição?	
Projeto de software: Sistema de interface entre o *kernel* e o seu hardware	
QS2: Qual a granularidade dos artefatos de software analisados para construir a predição?	
Módulo	
QS3: Que tipos de características estruturais são usados?	
Métricas de Código: Número de Operadores/Operandos Únicos, Número Total de Operadores/Operandos, Complexidade Ciclomática de McCabe Métricas de Dependência: Fan-In, Fan-Out, etc	
QS4: Para que propósito a abordagem é destinada?	
Predição de Qualidade	

QS5: Qual a técnica utilizada para construir a predição?
Categoria: Redes Neurais
QS6: Quais métricas de desempenho foram utilizadas para avaliar a abordagem?
Erro Absoluto
QS7: Qual é o desempenho aproximado das predições fornecido pela abordagem?
Média (Erro Absoluto) = 10000
QS8: Oferece apoio ferramental?
Não

Dados da Publicação	
Título	Investigating Arima Models Of Software System Quality
Autores	Khoshgoftaar T.M., Szabo R.M.
Ano da publicação	1995
Conferência	Software Quality Journal
Resumo	
O artigo mostra uma abordagem para a predição do número de falhas da próxima construção. Foi utilizado o modelo de predição ARIMA (*Autoregressive Integrated Moving Averages*), no qual se assume que predições são baseadas no histórico (série temporal) de observações (medidas) passadas.	
QS1: De onde são obtidos os dados para a predição?	
Projeto de software: Sistema de interface entre o *kernel* e o seu hardware	
QS2: Qual a granularidade dos artefatos de software analisados para construir a predição?	
Módulo	
QS3: Que tipos de características estruturais são usados?	
Métricas de Código: Número Total de Operadores/Operandos, Complexidade Ciclomática de McCabe Métricas de Dependência: Fan-In, Fan-Out, etc	
QS4: Para que propósito a abordagem é destinada?	
Predição de Defeitos/Falhas: número de falhas	
QS5: Qual a técnica utilizada para construir a predição?	
Categoria: Análise de Séries Temporais Nome: *Autoregressive Integrated Moving Averages* (ARIMA)	
QS6: Quais métricas de desempenho foram utilizadas para avaliar a abordagem?	
Erro Absoluto	
QS7: Qual é o desempenho aproximado das predições fornecido pela abordagem?	
Média (Erro Absoluto) = 5,1	
QS8: Oferece apoio ferramental?	
Não	

Dados da Publicação	
Título	Mining Static Code Metrics For A Robust Prediction Of Software Defect-Proneness
Autores	Lianfa Li; Leung, H.
Ano da publicação	2011
Conferência	Empirical Software Engineering and Measurement (ESEM), 2011 International Symposium on
Resumo	
O artigo mostra uma método (referido como *Robusto*) que seleciona, em uma etapa inicial, de um conjunto predefinido, algoritmos de predição que se mostraram mais qualificados e os integra, compondo, em seguida, um modelo de predição híbrido.	
QS1: De onde são obtidos os dados para a predição?	
Bases de medidas: NASA MDP	
QS2: Qual a granularidade dos artefatos de software analisados para construir a predição?	

Módulo
QS3: Que tipos de características estruturais são usados?
Métricas de Código: Métricas de Halstead e Métricas de McCabe
QS4: Para que propósito a abordagem é destinada?
Predição de Defeitos/Falhas: *Defect-proneness*
QS5: Qual a técnica utilizada para construir a predição?
Categoria: Composto Nome: *Integrative Meta-Learner*
QS6: Quais métricas de desempenho foram utilizadas para avaliar a abordagem?
Outros: Área Sob a Curva (AUC) ROC
QS7: Qual é o desempenho aproximado das predições fornecido pela abordagem?
Outros: AUC = 0,9
QS8: Oferece apoio ferramental?
Não

Dados da Publicação	
Título	Neural-Network Techniques For Software-Quality Evaluation
Autores	Kumar, R.; Rai, S.; Trahan, J.L.
Ano da publicação	1998
Conferência	Reliability and Maintainability Symposium, 1998. Proceedings., Annual
Resumo	
O artigo mostra uma abordagem que utiliza Redes Neurais, com o auxílio da Análise de Componentes Principais (PCA), para realizar predição da propensão a falhas. Métricas de complexidade são usadas como entrada para o algoritmo de predição.	
QS1: De onde são obtidos os dados para a predição?	
Projeto de software: Sistema Militar de Comando e Comunicações de Controle	
QS2: Qual a granularidade dos artefatos de software analisados para construir a predição?	
Módulo	
QS3: Que tipos de características estruturais são usados?	
Métricas de Código: Número de Operadores/Operandos Únicos, Número Total de Operadores/Operandos, Complexidade Ciclomática de McCabe, Complexidade Ciclomática Estendida, Número de Linhas de Código, etc	
QS4: Para que propósito a abordagem é destinada?	
Predição de Defeitos/Falhas: *Fault-proneness*	
QS5: Qual a técnica utilizada para construir a predição?	
Categoria: Redes Neurais Descrição: Redes Neurais, com auxílio da Análise de Componentes Principais (PCA)	
QS6: Quais métricas de desempenho foram utilizadas para avaliar a abordagem?	
Outros: Erro Tipo 1 (módulo com baixo risco de defeitos classificado como de alto risco), Erro Tipo 2 (módulo com alto risco de defeito classificado como de baixo risco)	
QS7: Qual é o desempenho aproximado das predições fornecido pela abordagem?	
Outros: Erro Tipo 2 = 14,28%	
QS8: Oferece apoio ferramental?	
Não	

Dados da Publicação	
Título	Practical Considerations In Deploying Ai For Defect Prediction: A Case Study Within The Turkish Telecommunication Industry
Autores	Tosun A., Turhan B., Bener A.
Ano da publicação	2009

Conferência	ACM International Conference Proceeding Series
Resumo	
O artigo mostra a execução de um projeto para realizar a análise e medição de um sistema de software de telecomunicações, a construção de um modelo *Naive Bayes* para predição de módulos propensos a defeitos antes da fase de teste e o armazenamento de um histórico de versões e bugs.	
QS1: De onde são obtidos os dados para a predição?	
Bases de medidas: NASA PROMISE Projeto de software: Sistema de informação de uma empresa de telecomunicações	
QS2: Qual a granularidade dos artefatos de software analisados para construir a predição?	
Módulo	
QS3: Que tipos de características estruturais são usados?	
Métricas de Código: Métricas de McCabe, Métricas de Halstead, Número de Linhas de Código Executáveis/Comentadas Métricas OO: Métricas de Chidamber-Kemerer	
QS4: Para que propósito a abordagem é destinada?	
Predição de Falhas	
QS5: Qual a técnica utilizada para construir a predição?	
Categoria: Classificação Nome: *Naive Bayes*	
QS6: Quais métricas de desempenho foram utilizadas para avaliar a abordagem?	
Cobetura, Taxa de Falsos Positivos	
QS7: Qual é o desempenho aproximado das predições fornecido pela abordagem?	
Cobertura = 89%	
QS8: Oferece apoio ferramental?	
Não	

Dados da Publicação	
Título	Predicting Class Libraries Interface Evolution: An Investigation Into Machine Learning Approaches
Autores	Sahraoui, H.A.; Boukadoum, A.M.; Lounis, H.; Etheve, F.
Ano da publicação	2000
Conferência	Software Engineering Conference, 2000. APSEC 2000. Proceedings. Seventh Asia-Pacific
Resumo	
O artigo apresenta uma abordagem baseada em lógica *Fuzzy* para construir um modelo de predição da estabilidade de interfaces de bibliotecas de classe orientadas a objeto.	
QS1: De onde são obtidos os dados para a predição?	
Projeto de software: Versões da biblioteca C++ chamada OSE	
QS2: Qual a granularidade dos artefatos de software analisados para construir a predição?	
Classe	
QS3: Que tipos de características estruturais são usados?	
Métricas OO: Depth of Inheritance Tree, Número de descendentes, Número de Ancestrais, Número de Métodos, Número de métodos herdados	
QS4: Para que propósito a abordagem é destinada?	
Predição de Qualidade: Estabilidade de interfaces de bibliotecas de classes	
QS5: Qual a técnica utilizada para construir a predição?	
Categoria: Classificação Descrição: Árvore de Decisão baseada em Lógica *Fuzzy*	
QS6: Quais métricas de desempenho foram utilizadas para avaliar a abordagem?	
Acurácia	
QS7: Qual é o desempenho aproximado das predições fornecido pela abordagem?	

Indisponível
QS8: Oferece apoio ferramental?
Não

Dados da Publicação	
Título	The Impact Of Design Properties On Development Cost In Object-Oriented Systems
Autores	Briand, L.C.; Wust, J.
Ano da publicação	2001
Conferência	Software Metrics Symposium, 2001. METRICS 2001. Proceedings. Seventh International
Resumo	
O artigo apresenta uma abordagem para predição de custos de desenvolvimento. Essa abordagem utiliza um algoritmo híbrido formado pelos algoritmos *Poisson Regression* e *CART (Classification and Regression Tree) Regression Tree*.	
QS1: De onde são obtidos os dados para a predição?	
Projeto de software: Editor de música LIOO	
QS2: Qual a granularidade dos artefatos de software analisados para construir a predição?	
Classe	
QS3: Que tipos de características estruturais são usados?	
Métricas OO: Número de Métodos (Implementados, Herdados e Total), Número de Atributos Não Herdados, Número de Parâmetros Métricas de Dependência: Acoplamento entre Classes, Lack of Cohesion in Methods	
QS4: Para que propósito a abordagem é destinada?	
Predição de Custos: Custos de desenvolvimento	
QS5: Qual a técnica utilizada para construir a predição?	
Categoria: Composta Descrição: Híbrido de *Poisson Regression* com *Regression*.	
QS6: Quais métricas de desempenho foram utilizadas para avaliar a abordagem?	
Erro Relativo	
QS7: Qual é o desempenho aproximado das predições fornecido pela abordagem?	
Média (Erro Relativo) = 13%	
QS8: Oferece apoio ferramental?	
Não	

Dados da Publicação	
Título	Predicting Effectiveness Of Automatic Testing Tools
Autores	Daniel, B.; Boshernitsan, M.
Ano da publicação	2008
Conferência	Automated Software Engineering, 2008. ASE 2008. 23rd IEEE/ACM International Conference on
Resumo	
O artigo apresenta uma técnica que realiza, através do algoritmo C4.5, a predição da efetividade de uma ferramenta, medida em termos de cobertura (de testes), usando métricas derivadas da estrutura do programa.	
QS1: De onde são obtidos os dados para a predição?	
Projeto de software	
QS2: Qual a granularidade dos artefatos de software analisados para construir a predição?	
Método	
QS3: Que tipos de características estruturais são usados?	
Métricas de Código: Método Público [booleano], Método Privado [booleano], Complexidade Ciclomática, Número de Referências a Parâmetros, Número de Tipos não primitivos usados no método, Número de Acessos a Campos Estáticos	
QS4: Para que propósito a abordagem é destinada?	
Outros: Predição do nível de cobertura de um método	

QS5: Qual a técnica utilizada para construir a predição?
Categoria: Classificação Nome: C4.5
QS6: Quais métricas de desempenho foram utilizadas para avaliar a abordagem?
Acurácia
QS7: Qual é o desempenho aproximado das predições fornecido pela abordagem?
Acurácia = 85%
QS8: Oferece apoio ferramental?
Não

Dados da Publicação	
Título	Predicting Mutation Score Using Source Code And Test Suite Metrics
Autores	Jalbert, K.; Bradbury, J.S.
Ano da publicação	2012
Conferência	Realizing Artificial Intelligence Synergies in Software Engineering (RAISE), 2012 First International Workshop on
Resumo	
O artigo apresenta uma abordagem que usa o algoritmo *Support Vector Machine* para predizer a pontuação de mutação (*mutation score*) baseada na combinação de métricas de código e de testes da unidade de compilação em análise. A pontuação de mutação é definida como a porcentagem de mutantes que são detectados pela suíte de testes. A abordagem proposta visa aumentar o desempenho e diminuir o custo dos processos de testes com uso de mutantes, já que o número de mutantes gerados para um pequeno programa pode chegar à magnitude de centenas ou milhares.	
QS1: De onde são obtidos os dados para a predição?	
Casos de teste: Casos de teste JUnit	
QS2: Qual a granularidade dos artefatos de software analisados para construir a predição?	
Classe, Método	
QS3: Que tipos de características estruturais são usados?	
Métricas OO: Número de Métodos Sobrescritos, Número de Atributos, Complexidade Ciclomática Total, Número de Métodos, Número de Filhos, etc. Métricas de Código: Número de Linhas de Código em Método, Complexidade Ciclomática, Profundidade de Aninhamento do Bloco, Número de Parâmetros Métricas de Dependência: Lack of Cohesion in Methods	
QS4: Para que propósito a abordagem é destinada?	
Outros: Predição da pontuação de mutação (*mutation score*)	
QS5: Qual a técnica utilizada para construir a predição?	
Categoria: Classificação Nome: *Support Vector Machine*	
QS6: Quais métricas de desempenho foram utilizadas para avaliar a abordagem?	
Acurácia	
QS7: Qual é o desempenho aproximado das predições fornecido pela abordagem?	
Acurácia = 56%	
QS8: Oferece apoio ferramental?	
Não	

Dados da Publicação	
Título	Predicting Qualitative Assessments Using Fuzzy Aggregation
Autores	Pizzi, N.J.; Pedrycz, W.
Ano da publicação	2006

Conferência	Fuzzy Information Processing Society, 2006. NAFIPS 2006. Annual meeting of the North American
Resumo	
O artigo apresenta uma abordagem para a predição do nível de complexidade e manutenibilidade. Utiliza a técnica *Fuzzy Aggregation*, a qual possibilita a agregação de vários algoritmos de predição, escolhendo o melhor deles para realizar a predição da instância requerida.	
QS1: De onde são obtidos os dados para a predição?	
Projeto de software: Sistema de Análise de Dados Biomédicos EvIdent	
QS2: Qual a granularidade dos artefatos de software analisados para construir a predição?	
Classe	
QS3: Que tipos de características estruturais são usados?	
Métricas de Código: Número Total de Linhas de Código, Número Médio de Tokens por método, Número Médio de Decisões por método, Porcentagem de Membros Privados, Métricas de Halstead, Número de Métodos Sobrescritos Métricas de Dependência: Lack of Cohesion in Methods	
QS4: Para que propósito a abordagem é destinada?	
Predição de Qualidade: nível de Complexidade e Manutenibilidade	
QS5: Qual a técnica utilizada para construir a predição?	
Categoria: Composto Nome: *Fuzzy Integral*	
QS6: Quais métricas de desempenho foram utilizadas para avaliar a abordagem?	
Acurácia	
QS7: Qual é o desempenho aproximado das predições fornecido pela abordagem?	
Acurácia = 80%	
QS8: Oferece apoio ferramental?	
Não	

Dados da Publicação	
Título	Predicting Testability Of Program Modules Using A Neural Network
Autores	Khoshgoftaar, T.M.; Allen, E.B.; Xu, Z.
Ano da publicação	2000
Conferência	Application-Specific Systems and Software Engineering Technology, 2000. Proceedings. 3rd IEEE Symposium on
Resumo	
O artigo aplica Redes Neurais para a predição do nível de testabilidade (em valor numérico) a partir de métricas estáticas de software.	
QS1: De onde são obtidos os dados para a predição?	
Projeto de software: Sistema de piloto automático do Boeing 737	
QS2: Qual a granularidade dos artefatos de software analisados para construir a predição?	
Módulo	
QS3: Que tipos de características estruturais são usados?	
Métricas de Código: Número de Operadores/Operandos Distintos, Número Total de Operadores/Operandos, Complexidade Ciclomática de McCabe, Número de Operadores Lógicos, etc	
QS4: Para que propósito a abordagem é destinada?	
Predição de Qualidade: Testabilidade	
QS5: Qual a técnica utilizada para construir a predição?	
Categoria: Redes Neurais	
QS6: Quais métricas de desempenho foram utilizadas para avaliar a abordagem?	
Erro Absoluto	
QS7: Qual é o desempenho aproximado das predições fornecido pela abordagem?	
Erro Absoluto = 0,1656	
QS8: Oferece apoio ferramental?	

Não

Dados da Publicação	
Título	Prediction Of Fault-Proneness At Early Phase In Object-Oriented Development
Autores	Kamiya, T.; Kusumoto, S.; Inoue, K.
Ano da publicação	1999
Conferência	Object-Oriented Real-Time Distributed Computing, 1999. (ISORC '99) Proceedings. 2nd IEEE International Symposium on

Resumo
O artigo propõe um novo método para predizer a propensão a falhas (*fault-proneness*) de uma classe em qualquer fase do processo de desenvolvimento do projeto. Em primeiro lugar, definem-se quatro *checkpoints* no processo de desenvolvimento de uma classe, cada um deles correspondendo a uma etapa (ou a um conjunto delas) desse processo. Em seguida, define-se um subconjunto de métricas convencionais aplicáveis ao que é produzido na(s) etapa(s) de cada *checkpoint*. Finalmente, em cada checkpoint, estima-se a propensão a falhas da classe usando a técnica de regressão *Multivariate Logistic Regression Analysis* com as métricas que se lhe aplicam. Dessa forma, predições podem ser obtidas ao longo de todo o processo de desenvolvimento.
QS1: De onde são obtidos os dados para a predição?
Não é especificado
QS2: Qual a granularidade dos artefatos de software analisados para construir a predição?
Classe
QS3: Que tipos de características estruturais são usados?
Métricas OO: Complexidade Ciclomática Total, Número de Variáveis de Instância Métricas de Dependência: Acoplamento entre Classes, Falta de Coesão em Métodos
QS4: Para que propósito a abordagem é destinada?
Predição de Defeitos/Falhas: *Fault-proneness*
QS5: Qual a técnica utilizada para construir a predição?
Categoria: Regressão Nome: *Multivariate Logistic Regression Analysis*
QS6: Quais métricas de desempenho foram utilizadas para avaliar a abordagem?
Precisão, Cobertura
QS7: Qual é o desempenho aproximado das predições fornecido pela abordagem?
Precisão = 83%
QS8: Oferece apoio ferramental?
Não

Dados da Publicação	
Título	Prioritizing Software Security Fortification Through Code-Level Metrics
Autores	Gegick M., Williams L., Osborne J., Vouk M.
Ano da publicação	2008
Conferência	Proceedings of the ACM Conference on Computer and Communications Security

Resumo
O artigo propõe modelos para a predição da propensão a ataques (*attack-proneness*) em componentes de software.
QS1: De onde são obtidos os dados para a predição?
Projeto de software: Sistema comercial de software de telecomunicações
QS2: Qual a granularidade dos artefatos de software analisados para construir a predição?
Componente
QS3: Que tipos de características estruturais são usados?
Métricas de Código: Número de Linhas de Código
QS4: Para que propósito a abordagem é destinada?

Predição de Ataques: *Attack-proneness*
QS5: Qual a técnica utilizada para construir a predição?
Categoria: Híbrido de Classificação e Regressão Nome: *Classification And Regression Tree*
QS6: Quais métricas de desempenho foram utilizadas para avaliar a abordagem?
Outros: Erro Tipo 1 (componente não propenso a ataques classificado como propenso a ataques), Erro Tipo 2 (compoenente propenso a ataques classificado como não propenso a ataques), Coeficiente de Múltipla Determinação
QS7: Qual é o desempenho aproximado das predições fornecido pela abordagem?
Outros: Erro Tipo 2 = 0%
QS8: Oferece apoio ferramental?
Não

Dados da Publicação	
Título	Reducing Corrective Maintenance Effort Considering Module's History
Autores	Pighin, M.; Marzona, A.
Ano da publicação	2005
Conferência	Software Maintenance and Reengineering, 2005. CSMR 2005. Ninth European Conference on
Resumo	
O artigo apresenta uma abordagem que utiliza a Complexidade Ciclomática de McCabe juntamente com um Fator de Persistência (derivado da idade do módulo) para predizer módulos como de Alto ou Baixo Risco (de causar falhas).	
QS1: De onde são obtidos os dados para a predição?	
Projeto de software: Sistema comercial de gerência	
QS2: Qual a granularidade dos artefatos de software analisados para construir a predição?	
Módulo	
QS3: Que tipos de características estruturais são usados?	
Métricas de Código: Complexidade Ciclomática de McCabe	
QS4: Para que propósito a abordagem é destinada?	
Predição de Defeitos/Falhas: *Fault-proneness*	
QS5: Qual a técnica utilizada para construir a predição?	
Categoria: Técnica específica Descrição: Uma medida de risco $R_{j,k}$ de um módulo j em sua k-ésima release, na versão corrente do projeto, é calculada como o produto entre sua Complexidade Ciclomática de McCabe (C_j) e o seu Fator de Persistência ($F_{j,k}$). Os módulos são particionados em duas classes: classe Alto Risco, se seu $R_{j,k}$ for maior que um determinado valor (*threshold*), ou Baixo Risco, se seu $R_{j,k}$ for menor ou igual a esse mesmo valor.	
QS6: Quais métricas de desempenho foram utilizadas para avaliar a abordagem?	
Acurácia	
QS7: Qual é o desempenho aproximado das predições fornecido pela abordagem?	
Acurácia = 92,7%	
QS8: Oferece apoio ferramental?	
Não	

Dados da Publicação	
Título	Data Mining For Predictors Of Software Quality
Autores	Khoshgoftaar T.M., Allen E.B., Jones W.D., Hudepohl J.P.
Ano da publicação	1999
Conferência	International Journal of Software Engineering and Knowledge Engineering
Resumo	
O artigo apresenta diretrizes para a extração de métricas de processo inovadoras das bases de dados de Gerência de Configuração e	

Gerência de Modificações durante a fase de pré-processamento (*data-reduction*) dos dados no processo de KDD. Também o artigo adapta o algoritmo *Classification And Regression Tree*, resultando em um modelo que prediz se módulos estão propensos (*fault-proneness*) a terem falhas ou não, após a entrega aos clientes.
QS1: De onde são obtidos os dados para a predição?
Projeto de software
QS2: Qual a granularidade dos artefatos de software analisados para construir a predição?
Módulo
QS3: Que tipos de características estruturais são usados?
Métricas de Código: Número de Linhas de Código, Número de Comandos de Controle, Número de Variáveis Globais, Número de Variáveis Distintas, etc
QS4: Para que propósito a abordagem é destinada?
Predição de Defeitos/Falhas: *Fault-proneness*
QS5: Qual a técnica utilizada para construir a predição?
Categoria: Híbrido de Classificação e Regressão Nome: *Classification And Regression Tree*
QS6: Quais métricas de desempenho foram utilizadas para avaliar a abordagem?
Outros: Erro Tipo 1, Erro Tipo 2
QS7: Qual é o desempenho aproximado das predições fornecido pela abordagem?
Outros: Erro Tipo 2 = 28%
QS8: Oferece apoio ferramental?
Não

Dados da Publicação	
Título	Using The Support Vector Machine As A Classification Method For Software Defect Prediction With Static Code Metrics
Autores	Gray D., Bowes D., Davey N., Sun Y., Christianson B.
Ano da publicação	2009
Conferência	Communications in Computer and Information Science
Resumo	
A seção do livro apresenta uma abordagem que utiliza o algoritmo *Support Vector Machine* para a predição de falhas de um conjunto de módulos contidos em bases de dados da NASA.	
QS1: De onde são obtidos os dados para a predição?	
Bases de medidas: NASA MDP	
QS2: Qual a granularidade dos artefatos de software analisados para construir a predição?	
Módulo	
QS3: Que tipos de características estruturais são usados?	
Métricas de Código: Complexidade Ciclomática, Número de Operadores/Operandos, Número de Linhas de Código, etc	
QS4: Para que propósito a abordagem é destinada?	
Predição de Falhas	
QS5: Qual a técnica utilizada para construir a predição?	
Categoria: Classificação Nome: Support Vector Machine (SVM)	
QS6: Quais métricas de desempenho foram utilizadas para avaliar a abordagem?	
Acurácia	
QS7: Qual é o desempenho aproximado das predições fornecido pela abordagem?	
Acurácia = 70%	
QS8: Oferece apoio ferramental?	
Não	

APÊNDICE B – LISTA COMPLETA DAS PUBLICAÇÕES RETORNADAS PELA EXPRESSÃO DE BUSCA

Title	Authors	Year	Publication Title	1º filtro	2º filtro
A Clustering Algorithm For Software Fault Prediction	Kaur, D.; Kaur, A.; Gulati, S.; Aggarwal, M.	2010	Computer and Communication Technology (ICCCT), 2010 International Conference on	OK	OK
A Comparative Study Of Predictive Models For Program Changes During System Testing And Maintenance	Khoshgoftaar, T.M.; Munson, J.C.; Lanning, D.L.	1993	Software Maintenance ,1993. CSM-93, Proceedings., Conference on	OK	OK
A Density Based Clustering Approach For Early Detection Of Fault Prone Modules	Sandhu, P.S.; Kaur, M.; Kaur, A.	2010	Electronics and Information Engineering (ICEIE), 2010 International Conference On	OK	OK
A Metric-Based Approach For Predicting Conceptual Data Models Maintainability	Piattini M., Genero M., Jimenez L.	2001	International Journal of Software Engineering and Knowledge Engineering	OK	OK
A Model For Early Prediction Of Faults In Software Systems	Sandhu P.S., Brar A.S., Goel R., Kaur J., Anand S.	2010	2010 The 2nd International Conference on Computer and Automation Engineering, ICCAE 2010	OK	OK
A Model To Predict Anti-Regressive Effort In Open Source Software	Capiluppi, A.; Fernandez-Ramil, J.	2007	Software Maintenance, 2007. ICSM 2007. IEEE International Conference on	OK	OK
A Neural Network Approach For Predicting Software Development Faults	Khoshgoftaar, T.M.; Pandya, A.S.; More, H.B.	1992	Software Reliability Engineering, 1992. Proceedings., Third International Symposium on	OK	OK
A Novel Method For Early Software Quality Prediction Based On Support Vector Machine	Fei Xing; Ping Guo; Lyu, M.R.	2005	Software Reliability Engineering, 2005. ISSRE 2005. 16th IEEE International Symposium on	OK	OK
A Probabilistic Approach To Predict Changes In Object-Oriented Software Systems	Sharafat, Ali R.; Tahvildari, Ladan	2007	Software Maintenance and Reengineering, 2007. CSMR '07. 11th European Conference on	OK	OK
A Two-Step Model For Defect Density Estimation	Kutlubay, O.; Turhan, B.; Bener, A.B.	2007	Software Engineering and Advanced Applications, 2007. 33rd EUROMICRO Conference on	OK	OK
Adding Process Metrics To Enhance Modification Complexity Prediction	Toth, G.; Vegh, A.Z.; Beszedes, A.; Gyimothy, T.	2011	Program Comprehension (ICPC), 2011 IEEE 19th International Conference on	OK	OK
An Artificial Immune System Approach For Fault Prediction In Object-Oriented Software	Cata C., Diri B., Ozumut B.	2007	Proceedings - International Conference on Dependability of Computer Systems, DepCoS - RELCOMEX 2007	OK	OK
An Empirical Approach For Software Fault Prediction	Kaur, A.; Brar, A.S.; Sandhu, P.S.	2010	Industrial and Information Systems (ICIIS), 2010 International Conference on	OK	OK
Aode For Source Code Metrics For Improved Software Maintainability	Yingjie T., Chuanliang C., Chunhua Z.	2008	Proceedings of the 4th International Conference on Semantics, Knowledge, and Grid, SKG 2008	OK	OK
Application Of Neural Network For Predicting Software Development Faults Using Object-Oriented Design Metrics	Mie Mie Thet Thwin; Tong-Seng Quah	2002	Neural Information Processing, 2002. ICONIP '02. Proceedings of the 9th International Conference on	OK	OK
Automated Knowledge Acquisition And Application For Software Development Projects[27]	Baisch, E.; Liedtke, T.	1998	Automated Software Engineering, 1998. Proceedings. 13th IEEE International Conference on	OK	OK
Change Prediction In Object-Oriented Software Systems: A Probabilistic Approach[28]	Sharafat A.R., Tahvildari L.	2008	Journal of Software	OK	OK

[27] Trabalho similar a "Comparison Of Conventional Approaches And Soft-Computing Approaches For Software Quality Prediction". Não foi contabilizado.

Comparison Of Conventional Approaches And Soft-Computing Approaches For Software Quality Prediction	Baisch, E.; Liedtke, T.	1997	Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on	OK	OK
Design-Level Metrics Estimation Based On Code Metrics	Sami A., Fakhrahmad S.M.	2010	Proceedings of the ACM Symposium on Applied Computing	OK	OK
Does Calling Structure Information Improve The Accuracy Of Fault Prediction?	Shin Y., Bell R., Ostrand T., Weyuker E.	2009	Proceedings of the 2009 6th IEEE International Working Conference on Mining Software Repositories, MSR 2009	OK	OK
Early Software Fault Prediction Using Real Time Defect Data	Kaur, A.; Sandhu, P.S.; Bra, A.S.	2009	Machine Vision, 2009. ICMV '09. Second International Conference on	OK	OK
Evaluating Architectural Stability Using A Metric-Based Approach	Tonu S.A., Ashkan A., Tahvildari L.	2006	Proceedings of the European Conference on Software Maintenance and Reengineering, CSMR	OK	OK
Evaluating Web Applications Testability By Combining Metrics And Analogies	Marchetto, A.; Trentini, A.	2005	Information and Communications Technology, 2005. Enabling Technologies for the New Knowledge Society: ITI 3rd International Conference on	OK	OK
Gert: An Empirical Reliability Estimation And Testing Feedback Tool	Davidsson, M.; Zheng, J.; Nachiappan Nagappan; Williams, L.; Vouk, M.	2004	Software Reliability Engineering, 2004. ISSRE 2004. 15th International Symposium on	OK	OK
Improving Code Churn Predictions During The System Test And Maintenance Phases	Khoshgoftaar, T.M.; Szabo, R.M.	1994	Software Maintenance, 1994. Proceedings., International Conference on	OK	OK
Improving Neural Network Predictions Of Software Quality Using Principal Components Analysis	Khoshgoftaar Taghi M., Szabo Robert M.	1994	IEEE International Conference on Neural Networks - Conference Proceedings	OK	OK
Investigating Arima Models Of Software System Quality	Khoshgoftaar T.M., Szabo R.M.	1995	Software Quality Journal	OK	OK
Mining Static Code Metrics For A Robust Prediction Of Software Defect-Proneness	Lianfa Li; Leung, H.	2011	Empirical Software Engineering and Measurement (ESEM), 2011 International Symposium on	OK	OK
Neural-Network Techniques For Software-Quality Evaluation	Kumar, R.; Rai, S.; Trahan, J.L.	1998	Reliability and Maintainability Symposium, 1998. Proceedings., Annual	OK	OK
On The Use Of Calling Structure Information To Improve Fault Prediction	Shin Y., Bell R.M., Ostrand T.J., Weyuker E.J.	2012	Empirical Software Engineering	OK	OK
Practical Considerations In Deploying Ai For Defect Prediction: A Case Study Within The Turkish Telecommunication Industry	Tosun A., Turhan B., Bener A.	2009	ACM International Conference Proceeding Series	OK	OK
Practical Considerations In Deploying Statistical Methods For Defect Prediction: A Case Study Within The Turkish Telecommunications Industry[29]	Tosun A., Bener A., Turhan B., Menzies T.	2010	Information and Software Technology	OK	OK
Predicting Class Libraries Interface Evolution: An Investigation Into Machine Learning Approaches	Sahraoui, H.A.; Boukadoum, A.M.; Lounis, H.; Etheve, F.	2000	Software Engineering Conference, 2000. APSEC 2000. Proceedings. Seventh Asia-Pacific	OK	OK
The Impact Of Design Properties On Development Cost In Object-Oriented Systems	Briand, L.C.; Wust, J.	2001	Software Metrics Symposium, 2001. METRICS 2001. Proceedings. Seventh International	OK	OK
Predicting Effectiveness Of Automatic Testing Tools	Daniel, B.; Boshernitsan, M.	2008	Automated Software Engineering, 2008. ASE 2008. 23rd IEEE/ACM International Conference on	OK	OK

[28] Trabalho similar a "A Probabilistic Approach To Predict Changes In Object-Oriented Software Systems". Não foi contabilizado.

[29] Trabalho similar a "Practical Considerations In Deploying Ai For Defect Prediction: A Case Study Within The Turkish Telecommunication Industry". Não foi contabilizado.

Predicting Mutation Score Using Source Code And Test Suite Metrics	Jalbert, K.; Bradbury, J.S.	2012	Realizing Artificial Intelligence Synergies in Software Engineering (RAISE), 2012 First International Workshop on	OK	OK
Predicting Qualitative Assessments Using Fuzzy Aggregation	Pizzi, N.J.; Pedrycz, W.	2006	Fuzzy Information Processing Society, 2006. NAFIPS 2006. Annual meeting of the North American	OK	OK
Predicting Testability Of Program Modules Using A Neural Network	Khoshgoftaar, T.M.; Allen, E.B.; Xu, Z.	2000	Application-Specific Systems and Software Engineering Technology, 2000. Proceedings. 3rd IEEE Symposium on	OK	OK
Prediction Of Fault-Proneness At Early Phase In Object-Oriented Development	Kamiya, T.; Kusumoto, S.; Inoue, K.	1999	Object-Oriented Real-Time Distributed Computing, 1999. (ISORC '99) Proceedings. 2nd IEEE International Symposium on	OK	OK
Prioritizing Software Security Fortification Through Code-Level Metrics	Gegick M., Williams L., Osborne J., Vouk M.	2008	Proceedings of the ACM Conference on Computer and Communications Security	OK	OK
Reducing Corrective Maintenance Effort Considering Module's History	Pighin, M.; Marzona, A.	2005	Software Maintenance and Reengineering, 2005. CSMR 2005. Ninth European Conference on	OK	OK
Release Date Prediction For Telecommunication Software Using Bayesian Belief Networks	Ying Wang; Smith, M.	2002	Electrical and Computer Engineering, 2002. IEEE CCECE 2002. Canadian Conference on	OK	OK
Software Change Classification Using Hunk Metrics	Ferzund J., Ahsan S.N., Wotawa F.	2009	IEEE International Conference on Software Maintenance, ICSM	OK	OK
Software Quality Prediction Using Mixture Models With Em Algorithm	Ping Guo; Lyu, M.R.	2000	Quality Software, 2000. Proceedings. First Asia-Pacific Conference on	OK	OK
Software Science Model Of Compile Time	Shaw Jr. Wade H., Howatt James W., Maness Robert S., Miller Dennis M.	1992	IEEE Transactions on Software Engineering	OK	OK
Squaner: A Framework For Monitoring The Quality Of Software Systems	Haderer, N.; Khomh, F.; Antoniol, G.	2010	Software Maintenance (ICSM), 2010 IEEE International Conference on	OK	OK
Statistically-Based Program Size Estimation	Takahashi, M.; Miyake, T.; Hanata, S.	1989	Computer Software and Applications Conference, 1989. COMPSAC 89., Proceedings of the 13th Annual International	OK	OK
Using Statistical Models To Predict Software Regressions	Tarvo, A.	2008	Software Reliability Engineering, 2008. ISSRE 2008. 19th International Symposium on	OK	OK
Utilizing Computational Intelligence In Estimating Software Readiness	Tong Seng Quah; Mie Mie Thet Thwin	2006	Neural Networks, 2006. IJCNN '06. International Joint Conference on	OK	OK
Validation Of Network Measures As Indicators Of Defective Modules In Software Systems	Tosun A., Turhan B., Bener A.	2009	ACM International Conference Proceeding Series	OK	OK
Software Testing: A Machine Learning Experiment	Cheatham Thomas J., Yoo Jungsoon P., Wahl Nancy J.	1995	Proceedings - ACM Computer Science Conference	OK	OK
Tempest: Towards Early Identification Of Failure-Prone Binaries	Bhat, T.; Nagappan, N.	2008	Dependable Systems and Networks With FTCS and DCC, 2008. DSN 2008. IEEE International Conference on	OK	OK
Data Mining For Predictors Of Software Quality	Khoshgoftaar T.M., Allen E.B., Jones W.D., Hudepohl J.P.	1999	International Journal of Software Engineering and Knowledge Engineering	OK	OK
Using The Support Vector Machine As A Classification Method For Software Defect Prediction With Static Code Metrics	Gray D., Bowes D., Davey N., Sun Y., Christianson B.	2009	Communications in Computer and Information Science	OK	OK
Estimation Of Program Reverse Semantic Traceability Influence At Program Reliability With Assistance Of Object-Oriented Metrics	Vladimirovich R.V., Vladimirovich T.A., Dmitrievich K.A.	2009	2009 5th Central and Eastern European Software Engineering Conference in Russia, CEE-SECR 2009	OK	CE4
A Uml Approximation Of Three Chidamber-Kemerer Metrics And Their Ability To Predict Faulty Code Across Software Projects	Camargo Cruz A.E., Ochimizu K.	2010	IEICE Transactions on Information and Systems	OK	CE3

Association Of Program Size With Indicators Of Complexity	Davis J.Steve	1988	Cybernetics and Systems	OK	CE3
Bug Forecast: A Method For Automatic Bug Prediction	Ferenc R.	2010	Communications in Computer and Information Science	OK	CE3
Prest: An Intelligent Software Metrics Extraction, Analysis And Defect Prediction Tool	Kocaguneli E., Tosun A., Bener A., Turhan B., Cglayan B.	2009	Proceedings of the 21st International Conference on Software Engineering and Knowledge Engineering, SEKE 2009	OK	CE3
Refactoring Prediction Using Class Complexity Metrics	Kosker Y., Turhan B., Bener A.	2008	ICSOFT 2008 - Proceedings of the 3rd International Conference on Software and Data Technologies	OK	CE3
Application Of Neural Networks For Software Quality Prediction Using Object-Oriented Metrics	Tong-Seng Quah; Mie Mie Thet Thwin	2003	Software Maintenance, 2003. ICSM 2003. Proceedings. International Conference on	OK	CE2
A Proposal Of Nhpp-Based Method For Predicting Code Change In Open Source Development	Aman, H.	2011	Software Measurement, 2011 Joint Conference of the 21st Int'l Workshop on and 6th Int'l Conference on Software Process and Product Measurement (IWSM-MENSURA)	OK	CE1
A Change Impact Dependency Measure For Predicting The Maintainability Of Source Code	Xia F., Srikanth P.	2004	Proceedings - International Computer Software and Applications Conference	OK	CE1
A Reliability Model For Complex Systems	Schneidewind, N.; Hinchey, M.	2011	Secure Software Integration & Reliability Improvement Companion (SSIRI-C), 2011 5th International Conference on	OK	CE1
A Complexity Reliability Model	Schneidewind, N.; Hinchey, M.	2009	Software Reliability Engineering, 2009. ISSRE '09. 20th International Symposium on	OK	CE1
A Comparative Analysis Of The Efficiency Of Change Metrics And Static Code Attributes For Defect Prediction	Moser, R.; Pedrycz, W.; Succi, G.	2008	Software Engineering, 2008. ICSE '08. ACM/IEEE 30th International Conference on	OK	CE1
A Comparison Between Software Design And Code Metrics For The Prediction Of Software Fault Content	Zhao M., Wohlin C., Ohlsson N., Xie M.	1998	Information and Software Technology	OK	CE1
A Critique Of Software Defect Prediction Models	Benton N.E., Neil M.	1999	IEEE Transactions on Software Engineering	OK	CE1
A Metrics-Based Decision Support Tool For Software Module Interfacing Technique Selection To Lower Maintenance Cost	Bitman, W.R.	1999	Software Metrics Symposium, 1999. Proceedings. Sixth International	OK	CE1
A Multivariate Analysis Of Static Code Attributes For Defect Prediction	Turhan, B.; Bener, A.	2007	Quality Software, 2007. QSIC '07. Seventh International Conference on	OK	CE1
An Empirical Validation Of Object-Oriented Class Complexity Metrics And Their Ability To Predict Error-Prone Classes In Highly Iterative, Or Agile, Software: A Case Study	Olague H.M., Etzkorn L.H., Messimer S.L., Delugach H.S.	2008	Journal of Software Maintenance and Evolution	OK	CE1
An Investigation Of The Relationships Between Lines Of Code And Defects	Zhang H.	2009	IEEE International Conference on Software Maintenance, ICSM	OK	CE1
Applying Design Metrics To Predict Fault-Proneness: A Case Study On A Large-Scale Software System	Wong W.E., Horgan J.R., Syring M., Zage W., Zage D.	2000	Software - Practice and Experience	OK	CE1
Assessing Software Complexity From Uml Using Fractal Complexity Measure	V. Podgorelec; M. Herieko; M.B. Juric	2004	Computational Cybernetics, 2004. ICCC 2004. Second IEEE International Conference on	OK	CE1
Assessing The Capability Of Internal Metrics As Early Indicators Of Maintenance Effort Through Experimentation	Bocco M.G., Moody D.L., Piattini M.	2005	Journal of Software Maintenance and Evolution	OK	CE1
Assessing The Maintainability Of Software Product Line Feature Models Using Structural Metrics	Bagheri E., Gasevic D.	2011	Software Quality Journal	OK	CE1
Assessing Uml Design Metrics For Predicting Fault-Prone Classes In A	Nugroho A., Chaudron M.R.V., Arisholm E.	2010	Proceedings - International Conference on Software Engineering	OK	CE1

Java System					
Behavioral Dependency Measurement For Change-Proneness Prediction In Uml 2.0 Design Models	Han A.-R., Jeon S.-U., Bae D.-H., Hong J.-E.	2008	Proceedings - International Computer Software and Applications Conference	OK	CE1
Can Traditional Fault Prediction Models Be Used For Vulnerability Prediction?	Shin Y., Williams L.	2011	Empirical Software Engineering	OK	CE1
Can We Predict Types Of Code Changes? An Empirical Analysis	Giger, E.; Pinzger, M.; Gall, H.C.	2012	Mining Software Repositories (MSR), 2012 9th IEEE Working Conference on	OK	CE1
Code-Based Analysis Of The Development Effort Of A Large Scale Courseware Project	Marshall I.M., Price S., Dugard P.I., Hobbs P., Samson W.B.	1997	Information and Software Technology	OK	CE1
Comparing Design And Code Metrics For Software Quality Prediction	Jiang Y., Cukic B., Menzies T., Bartlow N.	2008	Proceedings - International Conference on Software Engineering	OK	CE1
Comparing Fine-Grained Source Code Changes And Code Churn For Bug Prediction	Giger E., Pinzger M., Gall H.C.	2011	Proceedings - International Conference on Software Engineering	OK	CE1
Comparison Of Web Size Measures For Predicting Web Design And Authoring Effort	Mendes, E.; Mosley, N.; Counsell, S.	2002	Software, IEE Proceedings -	OK	CE1
Complexity Measurement Of A Graphical Programming Language	Henry Sallie, Goff Roger	1989	Software - Practice and Experience	OK	CE1
Constructing Models For Predicting Extract Subclass Refactoring Opportunities Using Object-Oriented Quality Metrics	Al Dallal J.	2012	Information and Software Technology	OK	CE1
Crawlability Metrics For Automated Web Testing	Marchetto A., Tiella R., Tonella P., Alshahwan N., Harman M.	2011	International Journal on Software Tools for Technology Transfer	OK	CE1
Criticality Prediction Models Using Sdl Metrics Set	Euy-Seok Hong; Chi-Su Wu	1997	Software Engineering Conference, 1997. Asia Pacific ... and International Computer Science Conference 1997. APSEC '97 and ICSC '97. Proceedings	OK	CE1
Cyclomatic Complexity Density And Software Maintenance Productivity	Gill, G.K.; Kemerer, C.F.	1991	Software Engineering, IEEE Transactions on	OK	CE1
Design Evolution Metrics For Defect Prediction In Object Oriented Systems	Kpodjedo S., Ricca F., Galinier P., Gueheneuc Y.-G., Antoniol G.	2011	Empirical Software Engineering	OK	CE1
Different Strokes For Different Folks: A Case Study On Software Metrics For Different Defect Categories	Misirli A.T., Caglayan B., Miranskyy A.V., Bener A., Ruffolo N.	2011	Proceedings - International Conference on Software Engineering	OK	CE1
Empirical Design Bugs Prediction For Verification	Qi Guo; Tianshi Chen; Haihua Shen; Yunji Chen; Yue Wu; Weiwu Hu	2011	Design, Automation & Test in Europe Conference & Exhibition (DATE), 2011	OK	CE1
Empirical Study Of Complexity Metrics In Cobol Programs	Jung H.-W., Pivka M., Kim J.-Y.	2000	Journal of Systems and Software	OK	CE1
Empirical Validation Of Class Diagram Complexity Metrics	Genero, M.; Piattini, M.; Jimenez, L.	2001	Computer Science Society, 2001. SCCC 2001. Proceedings. XXI Internatinal Conference of the Chilean	OK	CE1
Estimation Of Defect Proneness Using Design Complexity Measurements In Object-Oriented Software	Selvarani, R.; Nair, T.R.G.; Prasad, V.K.	2009	2009 International Conference on Signal Processing Systems	OK	CE1
Evaluating Defect Prediction Approaches: A Benchmark And An Extensive Comparison	D'Ambros M., Lanza M., Robbes R.	2012	Empirical Software Engineering	OK	CE1
Finding "early" Indicators Of Uml Class Diagrams Understandability And Modifiability	Genero, M.; Piatini, M.; Manso, E.	2004	Empirical Software Engineering, 2004. ISESE '04. Proceedings. 2004 International Symposium on	OK	CE1
Graph-Based Analysis And Prediction For Software Evolution	Bhattacharya P., Iliofotou M., Neamtiu I., Faloutsos M.	2012	Proceedings - International Conference on Software Engineering	OK	CE1
High Level Quantitative	Meeuws R., Sigdel K.,	2008	Proceedings of the 2008	OK	CE1

Interconnect Estimation For Early Design Space Exploration	Yankova Y., Bertels K.		International Conference on Field-Programmable Technology, ICFPT 2008		
Mining Metrics To Predict Component Failures	Nagappan N., Ball T., Zeller A.	2006	Proceedings - International Conference on Software Engineering	OK	CE1
On Software Fault Prediction By Mining Software Complexity Data With Dynamically Filtered Training Sets	Podgorelec V.	2009	Proc. 9th WSEAS Int. Conf. Simulation, Modelling and Optimization, SMO '09, 5th WSEAS Int. Symp. Grid Computing, Proc. 5th WSEAS Int. Symp. Digital Libraries, Proc. 5th WSEAS Int. Symp. Data Mining	OK	CE1
Software Metrics And Microcode: A Case Study	Triantafyllos G., Vassiliadis S., Delgado-Frias J.G.	1996	Journal of Software Maintenance and Evolution	OK	CE1
Predicting Fault-Prone Modules Based On Metrics Transitions	Higo Y., Murao K., Kusumoto S., Inoue K.	2008	DEFECTS'08: 2008 International Symposium on Software Testing and Analysis - Proceedings of the 2008 Workshop on Defects in Large Software Systems 2008, DEFECTS'08	OK	CE1
Reconsideration Of Software Reliability Measurements	Shiyi Xu	2007	Asian Test Symposium, 2007. ATS '07. 16th	OK	CE1
Reliability Prediction And Estimation Of Prolog Programs	Azem, A.; Belli, F.; Jedrzejowicz, P.	1994	Reliability, IEEE Transactions on	OK	CE1
Testing And Reliability Of Logic Programs	Azem, A.; Belli, F.; Jack, O.; Jedrzejowicz, P.	1993	Software Reliability Engineering, 1993. Proceedings., Fourth International Symposium on	OK	CE1
Usage Of Multiple Prediction Models Based On Defect Categories	Caglayan B., Tosun A., Miranskyy A., Bener A., Ruffolo N.	2010	ACM International Conference Proceeding Series	OK	CE1
Using Complexity, Coupling, And Cohesion Metrics As Early Indicators Of Vulnerabilities	Chowdhury I., Zulkernine M.	2011	Journal of Systems Architecture	OK	CE1
Which Code Construct Metrics Are Symptoms Of Post Release Failures	Nagappan M., Murphy B., Vouk M.	2011	Proceedings - International Conference on Software Engineering	OK	CE1
Using Code Metrics To Predict Maintenance Of Legacy Programs: A Case Study	Polo, M.; Piattini, M.; Ruiz, F.	2001	Software Maintenance, 2001. Proceedings. IEEE International Conference on	OK	CE1
Empirical Evaluation Of Mixed-Project Defect Prediction Models	Turhan, B.; Tosun, A.; Bener, A.	2011	Software Engineering and Advanced Applications (SEAA), 2011 37th EUROMICRO Conference on	OK	CE1
Exploring Software Measures To Assess Program Comprehension	Feigenspan, J.; Apel, S.; Liebig, J.; Kastner, C.	2011	Empirical Software Engineering and Measurement (ESEM), 2011 International Symposium on	OK	CE1
Predicting Oss Trustworthiness On The Basis Of Elementary Code Assessment	Lavazza L., Morasca S., Taibi D., Tosi D.	2010	ESEM 2010 - Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement	OK	CE1
Predicting Software Development Errors Using Software Complexity Metrics	Khoshgoftaar, T.M.; Munson, J.C.	1990	Selected Areas in Communications, IEEE Journal on	OK	CE1
Software Defect Prediction Using Call Graph Based Ranking (CGBR) Framework	Turhan, B.; Kocak, G.; Bener, A.	2008	Software Engineering and Advanced Applications, 2008. SEAA '08. 34th Euromicro Conference	OK	CE1
Software Defect Prediction Using Static Code Metrics Underestimates Defect-Proneness	Gray, D.; Bowes, D.; Davey, N.; Yi Sun; Christianson, B.	2010	Neural Networks (IJCNN), The 2010 International Joint Conference on	OK	CE1
The Effect Of Granularity Level On Software Defect Prediction	Calikli, G.; Tosun, A.; Bener, A.; Celik, M.	2009	Computer and Information Sciences, 2009. ISCIS 2009. 24th International Symposium on	OK	CE1
Understanding The Impact Of Code And Process Metrics On Post-Release Defects: A Case Study On The Eclipse Project	Shihab E., Jiang Z.M., Ibrahim W.M., Adams B., Hassan A.E.	2010	ESEM 2010 - Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement	OK	CE1
A Validation Of Object-Oriented Design Metrics As Quality Indicators	Basili V.R., Briand L.C., Melo W.L.	1996	IEEE Transactions on Software Engineering	OK	CE1
An Initial Study On The Use Of Execution Complexity Metrics As Indicators Of Software Vulnerabilities	Shin Y., Williams L.	2011	Proceedings - International Conference on Software Engineering	OK	CE1

Analysis Of Software Quality Cost Modeling?s Industrial Applicability With Focus On Defect Estimation	Karg, L.M.; Beckhaus, A.	2008	Industrial Engineering and Engineering Management, 2008. IEEM 2008. IEEE International Conference on	OK	CE1
Building Uml Class Diagram Maintainability Prediction Models Based On Early Metrics	Genero, M.; Piattini, M.; Manso, E.; Cantone, G.	2003	Software Metrics Symposium, 2003. Proceedings. Ninth International	OK	CE1
Empirical Investigation Of Fault Prediction Capability Of Object Oriented Metrics Of Open Source Software	Singh, Pradeep; Verma, Shrish	2012	Computer Science and Software Engineering (JCSSE), 2012 International Joint Conference on	OK	CE1
Empirical Studies To Assess The Understandability Of Data Warehouse Schemas Using Structural Metrics	Serrano M.A., Calero C., Sahraoui H.A., Piattini M.	2008	Software Quality Journal	OK	CE1
Empirically Validating Software Metrics For Risk Prediction Based On Intelligent Methods	Zhihong Xu; Xin Zheng; Ping Guo	2006	Intelligent Systems Design and Applications, 2006. ISDA '06. Sixth International Conference on	OK	CE1
Enhancing Predictive Models Using Principal Component Analysis And Search Based Metric Selection: A Comparative Study	Vivanco R., Jin D.	2008	ESEM'08: Proceedings of the 2008 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement	OK	CE1
Evaluating Complexity, Code Churn, And Developer Activity Metrics As Indicators Of Software Vulnerabilities	Shin Y., Meneely A., Williams L., Osborne J.A.	2011	IEEE Transactions on Software Engineering	OK	CE1
Exploring Defect Data From Development And Customer Usage On Software Modules Over Multiple Releases	Biyani, S.; Santhanam, P.	1998	Software Reliability Engineering, 1998. Proceedings. The Ninth International Symposium on	OK	CE1
Fault Detection And Prediction In An Open-Source Software Project	English M., Exton C., Rigon I., Cleary B.	2009	ACM International Conference Proceeding Series	OK	CE1
Ineffectiveness Of Use Of Software Science Metrics As Predictors Of Defects In Object Oriented Software	Rana, Z.A.; Shamail, S.; Awais, M.M.	2009	Software Engineering, 2009. WCSE '09. WRI World Congress on	OK	CE1
Micro Interaction Metrics For Defect Prediction	Lee T., Han D.G., Kim S., In H.P.	2011	Proceedings of the ACM SIGSOFT Symposium on the Foundations of Software Engineering	OK	CE1
Mining Input Sanitization Patterns For Predicting Sql Injection And Cross Site Scripting Vulnerabilities	Lwin Khin Shar; Hee Beng Kuan Tan	2012	Software Engineering (ICSE), 2012 34th International Conference on	OK	CE1
Mining Software Metrics From Jazz	Finlay, J.; Connor, A.M.; Pears, R.	2011	Software Engineering Research, Management and Applications (SERA), 2011 9th International Conference on	OK	CE1
Modeling Class Cohesion As Mixtures Of Latent Topics	Yixun Liu; Poshyvanyk, D.; Ferenc, R.; Gyimothy, T.; Chrisochoides, N.	2009	Software Maintenance, 2009. ICSM 2009. IEEE International Conference on	OK	CE1
Network Versus Code Metrics To Predict Defects: A Replication Study	Premraj R., Herzig K.	2011	Proceedings - 2011 5th International Symposium on Empirical Software Engineering and Measurement, ESEM 2011	OK	CE1
New Conceptual Coupling And Cohesion Metrics For Object-Oriented Systems	U?jha?zi, B.; Ferenc, R.; Poshyvanyk, D.; Gyimo?thy, T.	2010	Source Code Analysis and Manipulation (SCAM), 2010 10th IEEE Working Conference on	OK	CE1
On The Ability Of Complexity Metrics To Predict Fault-Prone Classes In Object-Oriented Systems	Zhou Y., Xu B., Leung H.	2010	Journal of Systems and Software	OK	CE1
Parsimonious Classifiers For Software Quality Assessment	Miyoung Shin; Goel, A.L.; Ratanothayanon, S.; Paul, R.A.	2007	High Assurance Systems Engineering Symposium, 2007. HASE '07. 10th IEEE	OK	CE1
Predicting Coding Effort In Projects Containing Xml	Karus S., Dumas M.	2012	Proceedings of the European Conference on Software Maintenance and Reengineering, CSMR	OK	CE1
Predicting Defect Densities In Source Code Files With Decision Tree Learners	Knab P., Pinzger M., Bernstein A.	2006	Proceedings - International Conference on Software Engineering	OK	CE1

Predicting Defects Using Network Analysis On Dependency Graphs	Zimmermann, T.; Nagappan, N.	2008	Software Engineering, 2008. ICSE '08. ACM/IEEE 30th International Conference on	OK	CE1
Predicting Source-Code Complexity At The Design Stage	Henry, S.; Selig, C.	1990	Software, IEEE	OK	CE1
Using Source Code Metrics To Predict Change-Prone Java Interfaces	Romano, D.; Pinzger, M.	2011	Software Maintenance (ICSM), 2011 27th IEEE International Conference on	OK	CE1
Code Churn Estimation Using Organisational And Code Metrics: An Experimental Comparison	Karus S., Dumas M.	2012	Information and Software Technology	OK	CE1
Exploratory Study Of A Uml Metric For Fault Prediction	Camargo Cruz A.E.	2010	Proceedings - International Conference on Software Engineering	OK	CE1
Software Defect Prediction Using Bayesian Networks	Okutan A., Yildiz O.T.	2012	Empirical Software Engineering	OK	CE1
The Lines Of Code Metric As A Predictor Of Program Faults: A Critical Analysis	Khoshgoftaar, T.M.; Munson, J.C.	1990	Computer Software and Applications Conference, 1990. COMPSAC 90. Proceedings., Fourteenth Annual International	OK	CE1
Changes And Bugs - Mining And Predicting Development Activities	Zimmermann T.	2009	IEEE International Conference on Software Maintenance, ICSM	OK	CE1
Discovering Determinants Of High Volatility Software	Seaman, C.; Koru, A.G.; Sampath, S.	2009	Cooperative and Human Aspects on Software Engineering, 2009. CHASE '09. ICSE Workshop on	OK	CE1
Predicting Maintenance Performance Using Object-Oriented Design Complexity Metrics	Bandi, R.K.; Vaishnavi, V.K.; Turk, D.E.	2003	Software Engineering, IEEE Transactions on	OK	CE1
A Systematic And Comprehensive Investigation Of Methods To Build And Evaluate Fault Prediction Models	Arisholm E., Briand L.C., Johannessen E.B.	2010	Journal of Systems and Software	OK	CE1
Benchmarking Classification Models For Software Defect Prediction: A Proposed Framework And Novel Findings	Lessmann, S.; Baesens, B.; Mues, C.; Pietsch, S.	2008	Software Engineering, IEEE Transactions on	OK	CE1
Comparative Study Of Various Artificial Intelligence Techniques To Predict Software Quality	Khan, M.J.; Shamail, S.; Awais, M.M.; Hussain, T.	2006	Multitopic Conference, 2006. INMIC '06. IEEE	OK	CE1
Comparing Fault-Proneness Estimation Models	Bellini P., Bruno I., Nesi P., Rogai D.	2005	Proceedings of the IEEE International Conference on Engineering of Complex Computer Systems, ICECCS	OK	CE1
Data Mining Static Code Attributes To Learn Defect Predictors	Menzies, T.; Greenwald, J.; Frank, A.	2007	Software Engineering, IEEE Transactions on	OK	CE1
Data Mining Techniques For Building Fault-Proneness Models In Telecom Java Software	Arisholm, Erik; Briand, Lionel C.; Fuglerud, Magnus	2007	Software Reliability, 2007. ISSRE '07. The 18th IEEE International Symposium on	OK	CE1
Empirical Validation Of Object-Oriented Metrics On Open Source Software For Fault Prediction	Gyimothy T., Ferenc R., Siket I.	2005	IEEE Transactions on Software Engineering	OK	CE1
Naive Bayes Software Defect Prediction Model	Wang T., Li W.-H.	2010	2010 International Conference on Computational Intelligence and Software Engineering, CiSE 2010	OK	CE1
Predicting Defective Software Components From Code Complexity Measures	Hongyu Zhang; Xiuzhen Zhang; Ming Gu	2007	Dependable Computing, 2007. PRDC 2007. 13th Pacific Rim International Symposium on	OK	CE1
Predicting Defects In Sap Java Code: An Experience Report	Holschuh, T.; Pauser, M.; Herzig, K.; Zimmermann, T.; Premraj, R.; Zeller, A.	2009	Software Engineering - Companion Volume, 2009. ICSE-Companion 2009. 31st International Conference on	OK	CE1
Predicting Risky Modules In Open-Source Software For High-Performance Computing	Phadke A.A., Allen E.B.	2005	Proceedings - International Conference on Software Engineering	OK	CE1
Predictive Modeling Techniques Of Software Quality From Software Measures	Khoshgoftaar, T.M.; Munson, J.C.; Bhattacharya, B.B.; Richardson, G.D.	1992	Software Engineering, IEEE Transactions on	OK	CE1
Tree-Based Software Quality Estimation Models For Fault Prediction	Khoshgoftaar, T.M.; Seliya, N.	2002	Software Metrics, 2002. Proceedings. Eighth IEEE Symposium on	OK	CE1

Empirical Assessment Of Machine Learning Based Software Defect Prediction Techniques	Challagulla, V.U.B.; Bastani, F.B.; I-Ling Yen; Paul, R.A.	2005	Object-Oriented Real-Time Dependable Systems, 2005. WORDS 2005. 10th IEEE International Workshop on	OK	CE1
Cohesion Metrics For Predicting Maintainability Of Service-Oriented Software	Perepletchikov, M.; Ryan, C.; Frampton, K.	2007	Quality Software, 2007. QSIC '07. Seventh International Conference on	OK	CE1
Investigation Of Predictors Of Failures And Debugging Effort For Large Mis	Davis J.	1989	Information and Software Technology	OK	CE1
Investigation Of The Risk To Software Reliability And Maintainability Of Requirements Changes	Schneidewind, N.F.	2001	Software Maintenance, 2001. Proceedings. IEEE International Conference on	OK	CE1
Measuring Dynamic Program Complexity	Munson, J.C.; Khoshgoftaar, T.M.	1992	Software, IEEE	OK	CE1
Measuring The Complexity Of Component-Based System Architecture	Alhazbi, S.M.	2004	Information and Communication Technologies: From Theory to Applications, 2004. Proceedings. 2004 International Conference on	OK	CE1
Metrics For Targeting Candidates For Reuse: An Experimental Approach	Schach Stephen R., Yang Xuefeng	1995	Proceedings of the ACM Symposium on Applied Computing	OK	CE1
Optimizing And Simplifying Software Metric Models Constructed Using Maximum Likelihood Methods	Chan, V.K.Y.; Wong, W.E.	2005	Computer Software and Applications Conference, 2005. COMPSAC 2005. 29th Annual International	OK	CE1
Predicting Class Testability Using Object-Oriented Metrics	Bruntink, M.; van Deursen, A.	2004	Source Code Analysis and Manipulation, 2004. Fourth IEEE International Workshop on	OK	CE1
A New Method To Predict Software Defect Based On Rough Sets	Weimin Yang; Longshu Li	2008	Intelligent Networks and Intelligent Systems, 2008. ICINIS '08. First International Conference on	OK	CE1
Application Of A Statistical Methodology To Simplify Software Quality Metric Models Constructed Using Incomplete Data Samples	Chan, V.K.Y.; Wong, W.E.; Xie, T.F.	2006	Quality Software, 2006. QSIC 2006. Sixth International Conference on	OK	CE1
Bug Prediction Based On Fine-Grained Module Histories	Hata, H.; Mizuno, O.; Kikuno, T.	2012	Software Engineering (ICSE), 2012 34th International Conference on	OK	CE1
Regression Modelling Of Software Quality: Empirical Investigation	Munson J.C., Khoshgoftaar T.M.	1990	Information and Software Technology	OK	CE1
Towards Logistic Regression Models For Predicting Fault-Prone Code Across Software Projects	Cruz, A.; Ochimizu, K.	2009	Empirical Software Engineering and Measurement, 2009. ESEM 2009. 3rd International Symposium on	OK	CE1
Prediction Of Maintainability Using Software Complexity Analysis: An Extended Frt	Prasanth, N.N.; Ganesh, S.; Dalton, G.A.	2008	Computing, Communication and Networking, 2008. ICCCn 2008. International Conference on	OK	CE1
An Approach To Modelling Long-Term Growth Trends In Software Systems	Lehman, M.M.; Ramil, J.F.; Sandler, U.	2001	Software Maintenance, 2001. Proceedings. IEEE International Conference on	OK	CE1
Applications Of Fuzzy Logic To Software Metric Models For Development Effort Estimation	Gray, A.; MacDonell, S.	1997	Fuzzy Information Processing Society, 1997. NAFIPS '97., 1997 Annual Meeting of the North American	OK	CE1
A Consideration Of The Impact Of Interactions With Module Effects On The Direct Measurement Of Subjective Software Attributes	Moses, J.	2001	Software Metrics Symposium, 2001. METRICS 2001. Proceedings. Seventh International	OK	CE1
A Constructive Rbf Neural Network For Estimating The Probability Of Defects In Software Modules	Bezerra, M.E.R.; Oliveira, A.L.I.; Meira, S.R.L.	2007	Neural Networks, 2007. IJCNN 2007. International Joint Conference on	OK	CE1
A Fuzzy-Inference System Based Approach For The Prediction Of Quality Of Reusable Software Components	Singh Sandhu, P.; Singh, H.	2006	Advanced Computing and Communications, 2006. ADCOM 2006. International Conference on	OK	CE1
A Metric For Software Readability	Buse R.P.L., Weimer W.R.	2008	ISSTA'08: Proceedings of the 2008 International Symposium on Software Testing and Analysis 2008	OK	CE1

A Vector-Based Approach To Software Size Measurement And Effort Estimation	Hastings, T.E.; Sajeev, A.S.M.	2001	Software Engineering, IEEE Transactions on	OK	CE1
Active Monitoring For Control Systems Under Anticipatory Semantics	Changzhi Zhao; Wei Dong; Zhichang Qi	2010	Quality Software (QSIC), 2010 10th International Conference on	OK	CE1
An Approach To Predict Software Maintenance Cost Based On Ripple Complexity	Hirota, T.; Tohki, M.; Overstreet, C.M.; Hashimoto, M.; Cherinka, R.	1994	Software Engineering Conference, 1994. Proceedings., 1994 First Asia-Pacific	OK	CE1
Applications Of Information Theory To Software Engineering Measurement	Khoshgoftaar T.M., Allen E.B.	1994	Software Quality Journal	OK	CE1
Assembly Time Modeling Through Connective Complexity Metrics	Mathieson, J.L.; Wallace, B.A.; Summers, J.D.	2010	Manufacturing Automation (ICMA), 2010 International Conference on	OK	CE1
Backfiring: Converting Lines Of Code To Function Points	Jones, C.	1995	Computer	OK	CE1
Categorizing Software Applications For Maintenance	McMillan C., Linares-Vasquez M., Poshyvanyk D., Grechanik M.	2011	IEEE International Conference on Software Maintenance, ICSM	OK	CE1
Complexity Metric Of Data Enquiry Functions For Public Registers And Electronic Commerce	Strahonja, V.	2002	Information Technology Interfaces, 2002. ITI 2002. Proceedings of the 24th International Conference on	OK	CE1
Design Metrics For Web Application Maintainability Measurement	Ghosheh, E.; Black, S.; Qaddour, J.	2008	Computer Systems and Applications, 2008. AICCSA 2008. IEEE/ACS International Conference on	OK	CE1
Evaluating Algorithm Performance Metrics Tailored For Prognostics	Saxena, A.; Celaya, J.; Saha, B.; Saha, S.; Goebel, K.	2009	Aerospace conference, 2009 IEEE	OK	CE1
Extracting Facts From Open Source Software	Ferenc, R.; Siket, I.; Gyimothy, T.	2004	Software Maintenance, 2004. Proceedings. 20th IEEE International Conference on	OK	CE1
How Well Can Simple Metrics Represent The Performance Of Hpc Applications?	Carrington, L.C.; Laurenzano, M.; Snavely, A.; Campbell, R.L.; Davis, L.P.	2005	Supercomputing, 2005. Proceedings of the ACM/IEEE SC 2005 Conference	OK	CE1
Identifying High-Risk Scenarios Of Complex Systems Using Input Domain Partitioning	Cukic Bojan, Ammar Hany H., Lateef Khalid	1998	Proceedings of the International Symposium on Software Reliability Engineering, ISSRE	OK	CE1
Improving Predictive Models Of Software Quality Using An Evolutionary Computational Approach	Vivanco, R.	2007	Software Maintenance, 2007. ICSM 2007. IEEE International Conference on	OK	CE1
Information Entropy And Structural Metrics Based Estimation Of Situations As A Basis For Situation Awareness And Decision Support	Belkin, A.; Beyerer, J.	2012	Cognitive Methods in Situation Awareness and Decision Support (CogSIMA), 2012 IEEE International Multi-Disciplinary Conference on	OK	CE1
Information Flow Metrics And Complexity Measurement	Sarala, S.; Jabbar, P.A.	2010	Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on	OK	CE1
Learning A Metric For Code Readability	Buse, R.P.L.; Weimer, W.R.	2010	Software Engineering, IEEE Transactions on	OK	CE1
Measurement, Prediction And Risk Analysis For Web Applications	Fewster R., Mendes E.	2001	International Software Metrics Symposium, Proceedings	OK	CE1
On Measuring The Complexity Of An Estelle Specification	Huang S.J., Lai R.	1998	Journal of Systems and Software	OK	CE1
Predicting C++ Program Quality By Using Bayesian Belief Networks	Masoud, F.A.M.; Shaikh, M.U.; Rabab'ah, O.M.A.	2004	Information and Communication Technologies: From Theory to Applications, 2004. Proceedings. 2004 International Conference on	OK	CE1
Using The Conceptual Cohesion Of Classes For Fault Prediction In Object-Oriented Systems	Marcus, A.; Poshyvanyk, D.; Ferenc, R.	2008	Software Engineering, IEEE Transactions on	OK	CE1
Quality Assessment Based On Attribute Series Of Software Evolution	Ratzinger, J.; Gall, H.; Pinzger, M.	2007	Reverse Engineering, 2007. WCRE 2007. 14th Working Conference on	OK	CE1
Simplified Workload Characterization Using Unified Prediction		2000	Performance Analysis of Systems and Software, 2000. ISPASS. 2000 IEEE International Symposium on	OK	CE1
Software Metrics Enhance Test Data Generation And Productivity Measurement	Jabbar, A.; Subramani, S.	2011	Information and Communication Technologies (WICT), 2011 World Congress on	OK	CE1

Study The Impact Of Improving Source Code On Software Metrics	Zoubi, Q.; Alsmadi, I.; Abul-Huda, B.	2012	Computer, Information and Telecommunication Systems (CITS), 2012 International Conference on	OK	CE1
Towards Portable Metrics-Based Models For Software Maintenance Problems	Bakota, T.; Ferenc, R.; Gyimothy, T.; Riva, C.; Jianli Xu	2006	Software Maintenance, 2006. ICSM '06. 22nd IEEE International Conference on	OK	CE1
Using Information Retrieval Based Coupling Measures For Impact Analysis	Poshyvanyk D., Marcus A., Ferenc R., Gyimothy T.	2009	Empirical Software Engineering	OK	CE1
Vulnerability Discovery In Multi-Version Software Systems	Jinyoo Kim; Malaiya, Y.K.; Ray, I.	2007	High Assurance Systems Engineering Symposium, 2007. HASE '07. 10th IEEE	OK	CE1
Web Structural Metrics Evaluation	Alsmadi I., Al-Taani A.T., Zaid N.A.	2010	Proceedings - 3rd International Conference on Developments in eSystems Engineering, DeSE 2010	OK	CE1
Structural Performance Measure Of Evolutionary Testing Applied To Worst-Case Timing Of Real-Time Systems	Gross, H.-G.; Jones, B.F.; Eyres, D.E.	2000	Software, IEE Proceedings -	OK	CE1
System-Of-Systems: An Architectural Framework To Support Development Cost Strategies	Malone, P.; Wolfarth, L.	2012	Aerospace Conference, 2012 IEEE	OK	CE1
Using Influence Diagrams For Software Risk Analysis	Chee, C.L.; Vij, V.; Ramamoorthy, C.V.	1995	Tools with Artificial Intelligence, 1995. Proceedings., Seventh International Conference on	OK	CE1
A Second Replicated Quantitative Analysis Of Fault Distributions In Complex Software Systems	Galinac Grbac, T.; Runeson, P.; Huljenic, D.	2012	Software Engineering, IEEE Transactions on	OK	CE1
An Empirical Study Of Pre-Release Software Faults In An Industrial Product Line	Devine T.R., Goseva-Popstajanova K., Krishnan S., Lutz R.R., Li J.J.	2012	Proceedings - IEEE 5th International Conference on Software Testing, Verification and Validation, ICST 2012	OK	CE1
Predicting Testability Of Eclipse: A Case Study	Singh Y., Saha A.	2010	Journal of Software Engineering	OK	CE1
Prediction Of Run-Time Failures Using Static Product Quality Metrics	Binkley A.B., Schach S.R.	1998	Software Quality Journal	OK	CE1
Study Of The Applicability Of Complexity Measures.	Davis John Stephen, LeBlanc Richard J.	1988	IEEE Transactions on Software Engineering	OK	CE1
A Study Of The Applicability Of Complexity Measures	Davis, J.S.; LeBlanc, R.J.	1988	Software Engineering, IEEE Transactions on	OK	CE1
A Model-Based Framework For The Integration Of Software Metrics	Evanco W.M., Lacovara R.	1994	The Journal of Systems and Software	OK	CE1
A Combined Linear & Nonlinear Approach For Classification Of Epileptic Eeg Signals	Balli, T.; Palaniappan, R.	2009	Neural Engineering, 2009. NER '09. 4th International IEEE/EMBS Conference on	CE1	--
A Comparison Of Image Quality Models And Metrics Based On Human Visual Sensitivity	Mayache, A.; Eude, T.; Cherifi, H.	1998	Image Processing, 1998. ICIP 98. Proceedings. 1998 International Conference on	CE1	--
A Complexity Measure For New Product Development Projects	Schlick, C.M.; Beutner, E.; Duckwitz, S.; Licht, T.	2007	Engineering Management Conference, 2007 IEEE International	CE1	--
A Controlled Expeniment On The Impact Of Software Structure On Maintainability	Rombach, H.D.	1987	Software Engineering, IEEE Transactions on	CE1	--
A Framework For Determining The Satisfiability Of General Boolean Expressions	Holman, C.S.	1994	Circuits and Systems, 1994., Proceedings of the 37th Midwest Symposium on	CE1	--
A Framework For Understanding Conceptual Changes In Evolving Source Code	Gold, N.; Mohan, A.	2003	Software Maintenance, 2003. ICSM 2003. Proceedings. International Conference on	CE1	--
A High Quality Adjustable Complexity Motion Estimation Algorithm For Video Encoders	Shahid, M.; Rossholm, A.; Lovstrom, B.	2011	Image and Signal Processing (CISP), 2011 4th International Congress on	CE1	--
A Hybrid Video Coder Based On Extended Macroblock Sizes, Improved Interpolation, And Flexible Motion Representation	Karczewicz, M.; Peisong Chen; Joshi, R.L.; Xianglin Wang; Wei-Jung Chien; Panchal, R.; Reznik, Y.; Coban, M.; In Suk Chong	2010	Circuits and Systems for Video Technology, IEEE Transactions on	CE1	--

A Methodology For Constructing Maintainability Model Of Object-Oriented Design	Kiewkanya, M.; Jindasawat, N.; Muenchaisri, P.	2004	Quality Software, 2004. QSIC 2004. Proceedings. Fourth International Conference on	CE1	--
A Mobile Knowledge Management Decision Support System For Automatically Conducting An Electronic Business	Wen W., Chen Y.H., Pao H.H.	2008	Knowledge-Based Systems	CE1	--
A Model For Continuous Query Latencies In Data Streams	Baldoni R., Di Luna G.A., Firmani D., Lodi G.	2011	ACM International Conference Proceeding Series	CE1	--
A New Approach Of Rate-Quantization Modeling For Intra And Inter Frames In H.264 Rate Control	Hrarti, M.; Saadane, H.; Larabi, M.-C.; Tamtaoui, A.; Aboutajdine, D.	2009	Signal and Image Processing Applications (ICSIPA), 2009 IEEE International Conference on	CE1	--
A New Logic Synthesis, Exorbds	Muma, K.; Seok-Bum Ko	2005	Electrical and Computer Engineering, 2005. Canadian Conference on	CE1	--
A New Metric For Processor Allocation Schemes In Multiprocessor Systems	Roy, S.; Chaudhary, V.	1997	Performance, Computing, and Communications Conference, 1997. IPCCC 1997., IEEE International	CE1	--
A New Video Quality Predictor Based On Decoder Parameter Extraction	Rossholm A., Lovstrom B.	2008	SIGMAP 2008 - Proceedings of the International Conference on Signal Processing and Multimedia Applications	CE1	--
A Novel Macroblock Layer Rate Control For H.264/Avc	Heng Yang; Qing Wang	2007	Multimedia and Expo, 2007 IEEE International Conference on	CE1	--
A Novel Objective No-Reference Metric For Digital Video Quality Assessment	Fuzheng Yang; Shuai Wan; Yilin Chang; Hong Ren Wu	2005	Signal Processing Letters, IEEE	CE1	--
A Pair-Programming Experiment In A Non-Programming Course	Gehringer E.F.	2003	Proceedings of the Conference on Object-Oriented Programming Systems, Languages, and Applications, OOPSLA	CE1	--
A Physically Based Dmos Transistor Model Implemented In Spice For Advanced Power Ic Tcad	Yeonbae Chung; Burk, D.E.	1995	Power Semiconductor Devices and ICs, 1995. ISPSD '95. Proceedings of the 7th International Symposium on	CE1	--
A Priori Wirelength And Interconnect Estimation Based On Circuit Characteristics	Balachandran, S.; Bhatia, D.	2005	Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on	CE1	--
A Quality Driven Extension To The Qvt-Relations Transformation Language	Drago M.L., Ghezzi C., Mirandola R.	2011	Computer Science - Research and Development	CE1	--
A Quantitative Approach To Software Maintainability Prediction	Liang Ping	2010	Information Technology and Applications (IFITA), 2010 International Forum on	CE1	--
A Reduced Complexity No-Reference Artificial Neural Network Based Video Quality Predictor	Shahid, M.; Rossholm, A.; Lovstrom, B.	2011	Image and Signal Processing (CISP), 2011 4th International Congress on	CE1	--
A Simpler Model Of Software Readability	Posnett D., Hindle A., Devanbu P.	2011	Proceedings - International Conference on Software Engineering	CE1	--
A Software Metric For Logical Errors And Integration Testing Effort	Leach, R.J.; Coleman, D.M.	1997	Computer Assurance, 1997. COMPASS '97. 'Are We Making Progress Towards Computer Assurance?'. Proceedings of the 12th Annual Conference on	CE1	--
A Software Science Model Of Compile Time	Shaw, W.H., Jr.; Howatt, J.W.; Maness, R.S.; Miller, D.M.	1989	Software Engineering, IEEE Transactions on	CE1	--
A Statistical Approach To Quantifying Clutter In Hyperspectral Infrared Images	Fadiran, O.O.; Molnar, P.; Kaplan, L.M.	2006	Aerospace Conference, 2006 IEEE	CE1	--
A Study Of Applying Extended Pie Technique To Software Testability Analysis	Tsung-Han Tsai; Chin-Yu Huang; Jun-Ru Chang	2009	Computer Software and Applications Conference, 2009. COMPSAC '09. 33rd Annual IEEE International	CE1	--
A Study Of Growth And The Relaxation Of Elastic Strain In Mgo On Fe(001)	Vassent, J. L.; Dynna, M.; Marty, A.; Gilles, B.; Patrat, G.	1996	Journal of Applied Physics	CE1	--
A Survey On Failure Prediction Of Large-Scale Server Clusters	Zhenghua Xue; Xiaoshe Dong; Siyuan Ma; Weiqing Dong	2007	Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2007. SNPD 2007. Eighth ACIS	CE1	--

			International Conference on		
A Systematic Literature Review Of Actionable Alert Identification Techniques For Automated Static Code Analysis	Heckman S., Williams L.	2011	Information and Software Technology	CE1	--
A Tool For Automatically Gathering Object-Oriented Metrics	Brooks, C.L.; Buell, C.G.	1994	Aerospace and Electronics Conference, 1994. NAECON 1994., Proceedings of the IEEE 1994 National	CE1	--
A-Priori Wirelength And Interconnect Estimation Based On Ciruit Characteristics	Balachandran S., Bhatia D.	2003	International Workshop on System Level Interconnect Prediction	CE1	--
Accurate Bit Prediction For Intra-Only Rate Control	Ling T., Xin Y., Yu S., Sun S.	2009	Proceedings - 2009 IEEE International Conference on Multimedia and Expo, ICME 2009	CE1	--
Adaptive Initial Quantization Parameter Selection For H.264/Svc Rate Control	Yang J., Sun Y., Kline C.S., Sun S.	2010	Proceedings - 2010 IEEE International Conference on Intelligent Computing and Intelligent Systems, ICIS 2010	CE1	--
Adaptive Linear Prediction For Resource Estimation Of Video Decoding	Andreopoulos, Y.; van der Schaar, M.	2007	Circuits and Systems for Video Technology, IEEE Transactions on	CE1	--
Adaptive Time Warp Simulation Of Timed Petri Nets	Ferscha, A.	1999	Software Engineering, IEEE Transactions on	CE1	--
Addressing Data-Complexity For Imbalanced Data-Sets: A Preliminary Study On The Use Of Preprocessing For C4.5	Luengo J., Fernandez A., Herrera F., Garcia S.	2009	ISDA 2009 - 9th International Conference on Intelligent Systems Design and Applications	CE1	--
Adopting The Cognitive Complexity Measure For Business Process Models	Gruhn, V.; Laue, R.	2006	Cognitive Informatics, 2006. ICCI 2006. 5th IEEE International Conference on	CE1	--
Advanced Wireless Product Design From Components To Smart Antenna Systems	Buris, N.E.	2007	Antennas and Propagation, 2007. EuCAP 2007. The Second European Conference on	CE1	--
Algorithmic Development Of Effectiveness Prediction For System Of Systems	Jackson, D.; Sedrick, G.; Tayeb, K.	2009	System Theory, 2009. SSST 2009. 41st Southeastern Symposium on	CE1	--
Allocation Of Iptv Streams Over Broadband Wireless Through Fuzzy Logic Control	Razavi, R.; Fleury, M.; Ghanbari, M.; Sammak, H.	2009	Computers and Communications, 2009. ISCC 2009. IEEE Symposium on	CE1	--
An Approach For Constructing Sparse Kernel Classifier	Yuan Z., Qu Y., Yang Y., Zheng N.	2006	Proceedings - International Conference on Pattern Recognition	CE1	--
An Architectural Quality Assessment For Domain-Specific Software	Changjun Hu; Feng Jiao; Chongchong Zhao	2008	Computer Science and Software Engineering, 2008 International Conference on	CE1	--
An Empirical Examination Of The Reverse Engineering Process For Binary Files	Sutherland I., Kalb G.E., Blyth A., Mulley G.	2006	Computers and Security	CE1	--
An Empirical Study Of The Evolution Of Php Web Application Security	Doyle, M.; Walden, J.	2011	Security Measurements and Metrics (Metrisec), 2011 Third International Workshop on	CE1	--
An Empirical Study On The Relation Between Dependency Neighborhoods And Failures	Zimmerman, T.; Nagappan, N.; Herzig, K.; Premraj, R.; Williams, L.	2011	Software Testing, Verification and Validation (ICST), 2011 IEEE Fourth International Conference on	CE1	--
An Ep Algorithm For Learning Highly Interpretable Classifiers	Cano, A.; Zafra, A.; Ventura, S.	2011	Intelligent Systems Design and Applications (ISDA), 2011 11th International Conference on	CE1	--
An Evaluation Of Software Design Using The Demeter Tool	Al-Janabi, A.; Aspinwall, E.	1993	Software Engineering Journal	CE1	--
An Experiment On Subjective Evolvability Evaluation Of Object-Oriented Software: Explaining Factors And Interrater Agreement	Mantyla, M.V.	2005	Empirical Software Engineering, 2005. 2005 International Symposium on	CE1	--
An Experiment To Assess Task Complexity In The Supervision Of Networked Systems	Murray, J.; Yili Liu	1997	Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on	CE1	--
An Experimental Study Of Environmental Complexity As Seen	Yang G., Anderson G.T.	2011	Conference Proceedings - IEEE International Conference on	CE1	--

By Robots			Systems, Man and Cybernetics		
An Experimental Study Of Software Metrics For Real-Time Software	Jensen, H.A.; Vairavan, K.	1985	Software Engineering, IEEE Transactions on	CE1	--
An Implementation Of Metrics Extraction And Analyzer Tool	Quah T.-S.	2007	International Journal on Artificial Intelligence Tools	CE1	--
An Improved Rate Control Algorithm For H.264	Hongtao Yu; Zhiping Lin; Feng Pan	2005	Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on	CE1	--
An Improved Srs Document Based Software Complexity Estimation And Its Robustness Analysis	Ashish S., Kushwaha D.S.	2011	Communications in Computer and Information Science	CE1	--
An Industrial Case Study Of Classifier Ensembles For Locating Software Defects	Misirli A.T., Bener A.B., Turhan B.	2011	Software Quality Journal	CE1	--
An Information Prediction Method Integrating Soft Data With Hard Data	Ting Zhang; Yi Du	2010	Mechanical and Electronics Engineering (ICMEE), 2010 2nd International Conference on	CE1	--
An Information Theory-Based Approach For Quantitative Evaluation Of User Interface Complexity	Kang, H.G.; Seong, P.H.	1998	Nuclear Science, IEEE Transactions on	CE1	--
An Integrated Approach For Criticality Prediction	Ebert, C.; Liedtke, T.	1995	Software Reliability Engineering, 1995. Proceedings., Sixth International Symposium on	CE1	--
An Object Oriented Complexity Metric Based On Cognitive Weights	Misra, S.	2007	Cognitive Informatics, 6th IEEE International Conference on	CE1	--
Analysis Of Motion-Complexity And Robustness For Video Transmission	Ishwar, P.; Puri, R.; Majumdar, A.	2005	Wireless Networks, Communications and Mobile Computing, 2005 International Conference on	CE1	--
Analytical Hierarchy Process Approach To Rank Measures For Structural Complexity Of Conceptual Models	Hussain, T.; Tahir, A.S.; Awais, M.M.; Shamail, S.	2006	Multitopic Conference, 2006. INMIC '06. IEEE	CE1	--
Analyzing Object Models With Theory Of Innovative Solution	Goyal, S.B.; Goyal, A.; Sharma, P.; Singhal, N.	2012	Advanced Computing & Communication Technologies (ACCT), 2012 Second International Conference on	CE1	--
Applicability Of Qualitative Ecg Processing To Wearable Computing	Bogunovic, N.; Smuc, T.	2008	Medical Devices and Biosensors, 2008. ISSS-MDBS 2008. 5th International Summer School and Symposium on	CE1	--
Application Of The Normal Form Of Vector Fields To Predict Interarea Separation In Power Systems	Thapar, J.; Vittal, V.; Kliemann, W.; Fouad, A.A.	1997	Power Systems, IEEE Transactions on	CE1	--
Applying Novel Resampling Strategies To Software Defect Prediction	Pelayo, L.; Dick, S.	2007	Fuzzy Information Processing Society, 2007. NAFIPS '07. Annual Meeting of the North American	CE1	--
Approximate Expressions For The Variances Of Non-Randomized Error Estimators And Cod Estimators For The Discrete Histogram Rule	Ting Chen; Braga-Neto, U.	2010	Genomic Signal Processing and Statistics (GENSIPS), 2010 IEEE International Workshop on	CE1	--
Are The Principal Components Of Software Complexity Data Stable Across Software Products?	Khoshgoftaar, T.M.; Lanning, D.L.	1994	Software Metrics Symposium, 1994., Proceedings of the Second International	CE1	--
Assessing The Maintainability Benefits Of Design Restructuring Using Dependency Analysis	Leitch, R.; Stroulia, E.	2003	Software Metrics Symposium, 2003. Proceedings. Ninth International	CE1	--
Audiovisual Quality Estimation For Mobile Video Services	Ries, M.; Gardlo, B.	2010	Selected Areas in Communications, IEEE Journal on	CE1	--
Audiovisual Quality Fusion Based On Relative Multimodal Complexity	Junyong You; Korhonen, J.; Reiter, U.	2011	Image Processing (ICIP), 2011 18th IEEE International Conference on	CE1	--
Automated Performance Prediction Of Message-Passing Parallel Programs	Block, R.J.; Sarukkai, S.; Mehra, P.	1995	Supercomputing, 1995. Proceedings of the IEEE/ACM SC95 Conference	CE1	--

Automated Prediction Of Epileptic Seizures In Rats With Recurrence Quantification Analysis	Gaoxiang Ouyang; Lijuan Xie; Huanwen Chen; Xiaoli Li; Xinping Guan; Huihua Wu	2005	Engineering in Medicine and Biology Society, 2005. IEEE-EMBS 2005. 27th Annual International Conference of the	CE1	--
Automatically Detecting The Quality Of The Query And Its Implications In Ir-Based Concept Location	Haiduc, S.	2011	Automated Software Engineering (ASE), 2011 26th IEEE/ACM International Conference on	CE1	--
Benchmark Circuit Complexity Validation Using Binary Decision Diagram Characteristics	Mills B., Prasad P.W.C., Prasad V.C.	2006	2006 Innovations in Information Technology, IIT	CE1	--
Bmat - A Binary Matching Tool For Stale Profile Propagation	Wang Z., Pierce K., McFarling S.	2000	Journal of Instruction-Level Parallelism	CE1	--
Bug Classification Using Program Slicing Metrics	Kai Pan; Kim, S.; Whitehead, E.J., Jr.	2006	Source Code Analysis and Manipulation, 2006. SCAM '06. Sixth IEEE International Workshop on	CE1	--
Bug Tracking And Reliability Assessment System (BTRAS)	Singh V.B., Chaturvedi K.K.	2011	International Journal of Software Engineering and its Applications	CE1	--
Bugcrawler: Visualizing Evolving Software Systems	D'Ambros M., Lanza M.	2007	Proceedings of the European Conference on Software Maintenance and Reengineering, CSMR	CE1	--
Building A Verification Test Plan: Trading Brute Force For Finesse	Bacchini, F.; Malik, S.; Bergeron, J.; Foster, H.; Piziali, A.; Mitra, R.S.; Ahlschlager, C.; Stein, D.	2006	Design Automation Conference, 2006 43rd ACM/IEEE	CE1	--
Building Software Quality Classification Trees: Approach, Experimentation, Evaluation	Takahashi, R.; Muraoka, Y.; Nakamura, Y.	1997	PROCEEDINGS The Eighth International Symposium On Software Reliability Engineering	CE1	--
Cache Miss Equations: A Compiler Framework For Analyzing And Tuning Memory Behavior	Ghosh S., Martonosi M., Malik S.	1999	ACM Transactions on Programming Languages and Systems	CE1	--
Cache-Oblivious Dynamic Programming For Bioinformatics	Chowdhury, R.A.; Hai-Son Le; Ramachandran, V.	2010	Computational Biology and Bioinformatics, IEEE/ACM Transactions on	CE1	--
Calibrating Function Point Backfiring Conversion Ratios Using Neuro-Fuzzy Technique	Wong J., Ho D., Capretz L.F.	2008	International Journal of Uncertainty, Fuzziness and Knowlege-Based Systems	CE1	--
Catheter Ablation Outcome Prediction In Persistent Atrial Fibrillation Based On Spatio-Temporal Complexity Measures Of The Surface Ecg	Meo, M.; Zarzoso, V.; Meste, O.; Latcu, D.G.; Saoudi, N.	2011	Computing in Cardiology, 2011	CE1	--
Cdna Cloning And Bioinformatic Analysis Of Self-Incompatible S34-Allele From Chinese Pears	Yanling Zeng; Xiaofeng Tan; Dangquan Zhang; Xiaoyi Hu; Hongpeng Chen; Deyi Yuan; Lin Zhang	2009	Bioinformatics and Biomedical Engineering , 2009. ICBBE 2009. 3rd International Conference on	CE1	--
Cfd Analysis Of Electrostatic Fluid Accelerators For Forced Convection Cooling	Jewell-Larsen, N.E.; Hsu, C.P.; Krichtafovitch, I.; Montgomery, S.; Dibene, J.; Mamishev, A.	2008	Dielectrics and Electrical Insulation, IEEE Transactions on	CE1	--
Change Bursts As Defect Predictors	Nagappan N., Zeller A., Zimmermann T., Herzig K., Murphy B.	2010	Proceedings - International Symposium on Software Reliability Engineering, ISSRE	CE1	--
Channel And Source Considerations Of A Bit-Rate Reduction Technique For A Possible Wireless Communications System's Performance Enhancement	Ilk, H.G.; Tugac, S.	2005	Wireless Communications, IEEE Transactions on	CE1	--
Characteristics Of Internet Latency And Their Impact On Distance Prediction Accuracy	Changyou Xing; Ming Chen	2009	Communication Networks and Services Research Conference, 2009. CNSR '09. Seventh Annual	CE1	--
Classifying Software Changes: Clean Or Buggy?	Kim S., Whitehead Jr. E.J., Zhang Y.	2008	IEEE Transactions on Software Engineering	CE1	--
Clustering Source Code Files To Predict Change Propagation During Software Maintenance	Bailey M., Lin K.-I., Sherrell L.	2012	Proceedings of the Annual Southeast Conference	CE1	--
Cmos Rf Modeling For Ghz Communication Ic's	Jia-Jiunn Ou; Xiaodong Jin; Ma, I.; Chenming Hu; Gray,	1998	VLSI Technology, 1998. Digest of Technical Papers. 1998 Symposium	CE1	--

	P.R.		on		
Co-Citation & Co-Reference Concepts To Control Focused Crawler Exploration	Maimunah S., Widyantoro D.H., Kuspriyanto, Sastramihardja H.S.	2011	Proceedings of the 2011 International Conference on Electrical Engineering and Informatics, ICEEI 2011	CE1	--
Coding Complexity Prediction For H.264/Avc Rate Control	Zhou Y., Tian L.	2009	IEICE Transactions on Information and Systems	CE1	--
Combinatorial Probability And The Tightness Of Generalization Bounds	Vorontsov K.V.	2008	Pattern Recognition and Image Analysis	CE1	--
Coming Of Age In An Object-Oriented World	Booch, G.	1994	Software, IEEE	CE1	--
Comparative Study Of Cognitive Complexity Measures	Misra, S.; Akman, I.	2008	Computer and Information Sciences, 2008. ISCIS '08. 23rd International Symposium on	CE1	--
Comparing Simulations And Graphical Representations Of Complexities Of Benchmark And Large-Variable Circuits	Prasad, P.W.C.; Beg, A.; Singh, A.K.	2010	Education Technology and Computer (ICETC), 2010 2nd International Conference on	CE1	--
Comparison Research Of Two Typical Uml-Class-Diagram Metrics: Experimental Software Engineering	Tong Yi	2010	Computer Application and System Modeling (ICCASM), 2010 International Conference on	CE1	--
Complexity Measures For Rule-Based Programs	O'Neal, M.B.; Edwards, W.R., Jr.	1994	Knowledge and Data Engineering, IEEE Transactions on	CE1	--
Complexity Measures For Secure Service-Oriented Software Architectures	Yanguo Liu; Traore, I.	2007	Predictor Models in Software Engineering, 2007. PROMISE'07: ICSE Workshops 2007. International Workshop on	CE1	--
Complexity-Constrained Video Bitstream Shaping	Andreopoulos, Y.; van der Schaar, M.	2007	Signal Processing, IEEE Transactions on	CE1	--
Complexity-Measure-Based Sequential Hypothesis Testing For Real-Time Detection Of Lethal Cardiac Arrhythmias	Chen S.-W.	2007	Eurasip Journal on Advances in Signal Processing	CE1	--
Computation Of The Complexity Of Vector Quantizers By Affine Modeling	Seraco E.P., Gomes J.G.R.C.	2011	Signal Processing	CE1	--
Computationally Efficient Flann-Based Intelligent Stock Price Prediction System	Patra, J.C.; Thanh, N.C.; Meher, P.K.	2009	Neural Networks, 2009. IJCNN 2009. International Joint Conference on	CE1	--
Content Adaptive Fast Motion Estimation Based On Spatio-Temporal Homogeneity Analysis And Motion Classification	Nisar H., Malik A.S., Choi T.-S.	2012	Pattern Recognition Letters	CE1	--
Controlled Experiment On The Impact Of Software Structure On Maintainability.	Rombach H.Dieter	1987	IEEE Transactions on Software Engineering	CE1	--
Core Level X-Ray Photoelectron Spectroscopy Of Rbtiopo 4 And Chemical Bonding In Ktiopo 4 Crystal Family	Makukha V.K., Atuchin V.V., Kesler V.G.	2010	2010 IEEE 2nd Russia School and Seminar on Fundamental Problems of Micro/Nanosystems Technologies, MNST'2010	CE1	--
Core Level X-Ray Photoelectron Spectroscopy Of Rbtiopo<Inf>4</Inf> And Chemical Bonding In Ktiopo<Inf>4</Inf>, Crystal Family	Makukha, Vladimir K.; Atuchin, Viktor V.; Kesler, Valeriy G.	2010	Fundamental Problems of Micro/Nanosystems Technologies (MNST), 2010 IEEE 2nd Russia School and Seminar on	CE1	--
Correlations Between Internal Software Metrics And Software Dependability In A Large Population Of Small C/C++ Programs	van der Meulen, M.J.P.; Revilla, M.A.	2007	Software Reliability, 2007. ISSRE '07. The 18th IEEE International Symposium on	CE1	--
Cots Score: An Acceptance Methodology For Cots Software	Morris, A.T.	2000	Digital Avionics Systems Conference, 2000. Proceedings. DASC. The 19th	CE1	--
Criterion And Measurement Of Complexity Of Stock Market: From Chaos, Fractal To Complexity Degree	Feng-tao Liu	2007	Wireless Communications, Networking and Mobile Computing, 2007. WiCom 2007. International Conference on	CE1	--
Cross-Architecture Performance	Marin G., Mellor-Crummey	2004	Performance Evaluation Review	CE1	--

Predictions For Scientific Applications Using Parameterized Models	J.				
Cyclogen: Automatic, Functional-Level Test Generator	Ayari, B.; Kaminska, B.	1992	Test Symposium, 1992. (ATS '92), Proceedings., First Asian (Cat. No.TH0458-0)	CE1	--
Cyclomatic Complexity And The Year 2000	McCabe, T.	1996	Software, IEEE	CE1	--
Dag3: A Tool For Design And Analysis Of Applications For Multicore Architectures	Drago M.L., Bishop J.	2012	Proceedings of the ACM Symposium on Applied Computing	CE1	--
Data Characterization For Effective Prototype Selection	Mollineda R.A., Sanchez J.S., Sotoca J.M.	2005	Lecture Notes in Computer Science	CE1	--
Defect Association And Complexity Prediction By Mining Association And Clustering Rules	Karthik, R.; Manikandan, N.	2010	Computer Engineering and Technology (ICCET), 2010 2nd International Conference on	CE1	--
Defining And Validating Metrics For Navigational Models	Abrahao, S.; Condori-Fernandez, N.; Olsina, L.; Pastor, O.	2003	Software Metrics Symposium, 2003. Proceedings. Ninth International	CE1	--
Definition And Validation Of Metrics For Itsm Process Models	Brito e Abreu, F.; de Braganca V da Porciuncula, R.; Freitas, J.M.; Costa, J.C.	2010	Quality of Information and Communications Technology (QUATIC), 2010 Seventh International Conference on the	CE1	--
Degree Of Complexity: A Maximum Entropy Based Error-Measure For Learning Endeavours In Neural Networks Dolores De Groff	De Groff D.	1996	Cybernetica	CE1	--
Delayed Generalized Predictive Control	Gomma, H.W.; Owens, D.H.	1998	Control '98. UKACC International Conference on (Conf. Publ. No. 455)	CE1	--
Density Estimation For Analog/Rf Test Problem Solving	Mir, S.; Stratigopoulos, H.-G.; Bounceur, A.	2010	VLSI Test Symposium (VTS), 2010 28th	CE1	--
Deriving Complexity Information From A Formal Communication Protocol Specification	Huang S.-J.	1998	Software - Practice and Experience	CE1	--
Design Of Joint Source And Channel Trellis Waveform Coders.	Ayanoglu Ender, Gray Robert M.	1987	IEEE Transactions on Information Theory	CE1	--
Design Pattern Prediction Techniques: A Comparative Analysis	Aslin Jenila, P.S.; Ranjana, P.	2011	Nanoscience, Engineering and Technology (ICONSET), 2011 International Conference on	CE1	--
Design, Analysis And Loss Minimization Of A Fractional-Slot Concentrated Winding Ipm Machine For Traction Applications	Tangudu, J.K.; Jahns, T.M.; Bohn, T.P.	2011	Energy Conversion Congress and Exposition (ECCE), 2011 IEEE	CE1	--
Detecting Tangled Logic Structures In Vlsi Netlists	Jindal, T.; Alpert, C.J.; Jiang Hu; Zhuo Li; Gi Joon Nam; Winn, C.B.	2010	Design Automation Conference (DAC), 2010 47th ACM/IEEE	CE1	--
Developing Intentional Systems With The Practionist Framework	Morreale, V.; Bonura, S.; Francaviglia, G.; Centineo, F.; Puccio, M.; Cossentino, M.	2007	Industrial Informatics, 2007 5th IEEE International Conference on	CE1	--
Development And Application Of Composite Complexity Models And A Relative Complexity Metric In A Software Maintenance Environment	Sherif, J.S.; Hops, J.M.	1996	WESCON/96	CE1	--
Development Of A Multiple Regression Equation To Predict Judo Performance With The Help Of Selected Structural And Body Composition Variables	Shaw, D.; Kavanal, B.K.	1995	Engineering in Medicine and Biology Society, 1995 and 14th Conference of the Biomedical Engineering Society of India. An International Meeting, Proceedings of the First Regional Conference., IEEE	CE1	--
Diagnose Effective Evolutionary Prototype Selection Using An Overlapping Measure	Garcia S., Cano J.-R., BernadO-Mansilla E., Herrera F.	2009	International Journal of Pattern Recognition and Artificial Intelligence	CE1	--
Distributed Construction Of A Multi-Level Topology With Unpredictable Metric Values For Wireless Networks	Lessmann, J.; Krishnamurthy, A.	2007	Wireless and Mobile Computing, Networking and Communications, 2007. WiMOB 2007. Third IEEE International Conference on	CE1	--

Dynamic Analysis Of Flexible Supercavitating Vehicles Using Modal-Based Elements	Choi J.-Y., Ruzzene M., Bauchau O.A.	2004	Simulation	CE1	--
Dynamic Coupling Measurement Of Object Oriented Software Using Trace Events	Kavitha, A.; Shanmugam, A.	2008	Applied Machine Intelligence and Informatics, 2008. SAMI 2008. 6th International Symposium on	CE1	--
Dynamic Density: Measuring And Predicting Sector Complexity [ATC]	Kopardekar, P.; Magyarits, S.	2002	Digital Avionics Systems Conference, 2002. Proceedings. The 21st	CE1	--
Dynamic Testing Complexity Metric	Voas J.	1992	Software Quality Journal	CE1	--
Dynamic Testing Of Knowledge Bases Using The Heuristic Testing Approach	Miller L.A.	1990	Expert Systems With Applications	CE1	--
Early Prediction Of Hardware Complexity In Hll-To-Hdl Translation	Cilardo, A.; Durante, P.; Lofiego, C.; Mazzeo, A.	2010	Field Programmable Logic and Applications (FPL), 2010 International Conference on	CE1	--
Effect Of Temperature On The Steady-State Sensitivity Of Vacuum Radiation Detectors	Amdur, I.; Brown, N. L.	1949	Review of Scientific Instruments	CE1	--
Effects Of Channel Prediction For Transmit Antenna Selection With Maximal-Ratio Combining In Rayleigh Fading	Prakash, S.; McLoughlin, I.	2011	Vehicular Technology, IEEE Transactions on	CE1	--
Efficient Algorithms And Software For Detection Of Full-Length Ltr Retrotransposons	Kalyanaraman, A.; Aluru, S.	2005	Computational Systems Bioinformatics Conference, 2005. Proceedings. 2005 IEEE	CE1	--
Efficient Metrics And High-Level Synthesis For Dynamically Reconfigurable Logic	Meribout, M.; Motomura, M.	2004	Very Large Scale Integration (VLSI) Systems, IEEE Transactions on	CE1	--
Efficient Microarchitectural Vulnerabilities Prediction Using Boosted Regression Trees And Patient Rule Inductions	Bin Li; Lide Duan; Lu Peng	2010	Computers, IEEE Transactions on	CE1	--
Electronic Properties And Bonding Characteristics Of Aln:Ag Thin Film Nanocomposites	Lekka, Ch. E.; Patsalas, P.; Komninou, Ph.; Evangelakis, G. A.	2011	Journal of Applied Physics	CE1	--
Elements Of System Design Optimization In Service Quality Management	Anerousis, N.; Yixin Diao; Heching, A.	2010	Network Operations and Management Symposium (NOMS), 2010 IEEE	CE1	--
Empirical Evaluation Of Selected Algorithms For Complexity-Based Classification Of Software Modules And A New Model	Wang J.H., Bouguila N., Bdiri T.	2010	Studies in Computational Intelligence	CE1	--
Empirical Investigation Of A Novel Approach To Check The Integrity Of Software Engineering Measuring Processes	Skylar Lei; Smith, M.; Succi, G.	2000	Software Engineering, 2000. Proceedings of the 2000 International Conference on	CE1	--
Empirical Study Of Software Maintenance Tasks	Jorgensen Magne	1995	Journal of Software Maintenance	CE1	--
Enabling Innovation In High Technology Organizations With Fixed Centralized Organizational Structures	Sholes, E.C.; Barnett, T.; Utley, D.R.	2011	Aerospace Conference, 2011 IEEE	CE1	--
Engineering Semantic Evaluation Of Decision Rules	Arciszewski T.	1997	Journal of Intelligent and Fuzzy Systems	CE1	--
Enhance Rule Based Detection For Software Fault Prone Modules	Najadat H., Alsmadi I.	2012	International Journal of Software Engineering and its Applications	CE1	--
Ensemble Of Software Defect Predictors: A Case Study	Tosun A., Turhan B., Bener A.	2008	ESEM'08: Proceedings of the 2008 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement	CE1	--
Estimating Parallel Performance, A Skeleton-Based Approach	Lobachev O., Loogen R.	2010	Proceedings of the ACM SIGPLAN International Conference on Functional Programming, ICFP	CE1	--
Estimating The Fault Content Of Software Using The Fix-On-Fix Model	Christenson D.A., Huang S.T.	1996	Bell Labs Technical Journal	CE1	--
Evaluating The Correlation Between Software Defect And Design Coupling Metrics	Abuasad, A.; Alsmadi, I.M.	2012	Computer, Information and Telecommunication Systems (CITS), 2012 International Conference on	CE1	--

Evaluating The Specificity Of Text Retrieval Queries To Support Software Engineering Tasks	Haiduc S., Bavota G., Oliveto R., Marcus A., De Lucia A.	2012	Proceedings - International Conference on Software Engineering	CE1	--
Evaluation Of Speckle-Interferometry Descriptors To Measuring Drying-Of-Coatings	Blotta E., Ballarin V., Brun M., Rabal H.	2011	Signal Processing	CE1	--
Evaluation Of Techniques To Detect Significant Network Performance Problems Using End-To-End Active Network Measurements	Les Cottrell, R.; Logg, C.; Chhaparia, M.; Gngonev, M.; Haro, F.; Nazir, F.; Sandford, M.	2006	Network Operations and Management Symposium, 2006. NOMS 2006. 10th IEEE/IFIP	CE1	--
Experiment To Assess Task Complexity In The Supervision Of Networked Systems	Murray John, Liu Yili	1997	Proceedings of the IEEE International Conference on Systems, Man and Cybernetics	CE1	--
Experimental Analysis Of The Root Causes Of Performance Evaluation Results: A Backfilling Case Study	Feitelson, D.G.	2005	Parallel and Distributed Systems, IEEE Transactions on	CE1	--
Experimental Verification And Finite Element Analysis Of Short-Circuit Electromagnetic Force For Dry-Type Transformer	Hyun-Mo Ahn; Yeon-Ho Oh; Joong-Kyoung Kim; Jae-Sung Song; Sung-Chin Hahn	2012	Magnetics, IEEE Transactions on	CE1	--
Exploring How To Support Software Revision In Software Non-Intensive Projects Using Existing Techniques	Kaiya, H.; Hara, K.; Kobayashi, K.; Osada, A.; Kaijiri, K.	2011	Computer Software and Applications Conference Workshops (COMPSACW), 2011 IEEE 35th Annual	CE1	--
Extracting Structural Characteristics Of A Nonlinear Time Series Using Genetic Algorithms	Adamopoulos, A.V.; Likothanassis, S.D.; Georgopoulos, E.F.	1997	Intelligent Information Systems, 1997. IIS '97. Proceedings	CE1	--
Factors In Multimedia Project And Process Management - Australian Survey Findings	Hannington A., Reed K.	2007	Proceedings of the Australian Software Engineering Conference, ASWEC	CE1	--
Failure Avoidance Through Fault Prediction Based On Synthetic Transactions	Shatnawi, M.; Ripeanu, M.	2011	Cluster, Cloud and Grid Computing (CCGrid), 2011 11th IEEE/ACM International Symposium on	CE1	--
Failure Is A Four-Letter Word - A Parody In Empirical Research	Zeller A., Zimmermann T., Bird C.	2011	ACM International Conference Proceeding Series	CE1	--
Fast And Accurate Resource Conflict Simulation For Performance Analysis Of Multi-Core Systems	Stattelmann, S.; Bringmann, O.; Rosenstiel, W.	2011	Design, Automation & Test in Europe Conference & Exhibition (DATE), 2011	CE1	--
Fast Identification Of Error-Prone Patterns For Ldpc Codes Under Message Passing Decoding	Jing Lei; Wen Gao	2008	Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE	CE1	--
Fault Prediction Using Early Lifecycle Data	Jiang, Yue; Cukic, Bojan; Menzies, Tim	2007	Software Reliability, 2007. ISSRE '07. The 18th IEEE International Symposium on	CE1	--
Fault Severity In Models In Fault-Correction Activity	Lanning David L., Khoshgoftaar Taghi M.	1995	IEEE Transactions on Reliability	CE1	--
Feature Selection With Stochastic Complexity	Dom, B.; Niblack, W.; Sheinvald, J.	1989	Computer Vision and Pattern Recognition, 1989. Proceedings CVPR '89., IEEE Computer Society Conference on	CE1	--
Fiexpat: Flexible Extraction Of Sequential Patterns	Rolland, P.-Y.	2001	Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on	CE1	--
Filtering System Metrics For Minimal Correlation-Based Self-Monitoring	Munawar, M.A.; Miao Jiang; Reidemeister, T.; Ward, P.A.S.	2009	Self-Adaptive and Self-Organizing Systems, 2009. SASO '09. Third IEEE International Conference on	CE1	--
Fluid Rewards For A Stochastic Process Algebra	Tribastone, M.; Jie Ding; Gilmore, S.; Hillston, J.	2012	Software Engineering, IEEE Transactions on	CE1	--
Forecasting Run-Times Of Secure Two-Party Computation	Schroepfer A., Kerschbaum F.	2011	Proceedings of the 2011 8th International Conference on Quantitative Evaluation of Systems, QEST 2011	CE1	--
Forecasting Workload And Airspace Configuration With Neural Networks And Tree Search Methods	Gianazza D.	2010	Artificial Intelligence	CE1	--

Frame Complexity-Based Rate-Quantization Model For H.264/Avc Intraframe Rate Control	Xuan Jing; Lap-Pui Chau; Wan-Chi Siu	2008	Signal Processing Letters, IEEE	CE1	--
Gcpsensor: A Cpu Performance Tool For Grid Environments	Dong Guo; Liang Hu; Meng Zhang; Zhang, Z.	2005	Quality Software, 2005. (QSIC 2005). Fifth International Conference on	CE1	--
Getting Started On Metrics - Jet Propulsion Laboratory Productivity And Quality	Bush M.W.	1990	Proceedings - International Conference on Software Engineering	CE1	--
Heuristic Evaluation Of Expansions For Non-Linear Hierarchical Slow Feature Analysis	Escalante, A.N.B.; Wiskott, L.	2011	Machine Learning and Applications and Workshops (ICMLA), 2011 10th International Conference on	CE1	--
Heuristic Model Research On Decision Tree Algorithm	Fa-chao Li; Fei Guan	2009	Intelligent Interaction and Affective Computing, 2009. ASIA '09. International Asia Symposium on	CE1	--
Heuristic, Systematic, And Informational Regularization For Process Monitoring	Gribok A.V., Hines J.W., Urmanov A., Uhrig R.E.	2002	International Journal of Intelligent Systems	CE1	--
High-Level Area Prediction For Power Estimation	Nemani, M.; Najm, F.N.	1997	Custom Integrated Circuits Conference, 1997., Proceedings of the IEEE 1997	CE1	--
High-Performance Timing Simulation Of Embedded Software	Schnerr J., Bringmann O., Viehl A., Rosenstiel W.	2008	Proceedings - Design Automation Conference	CE1	--
Human Rademacher Complexity	Zhu X., Rogers T.T., Gibson B.R.	2009	Advances in Neural Information Processing Systems 22 - Proceedings of the 2009 Conference	CE1	--
Hyperspectral Remote Sensing Subpixel Object Detection Performance	Kerekes, J.P.	2011	Applied Imagery Pattern Recognition Workshop (AIPR), 2011 IEEE	CE1	--
Identifying Comprehension Bottlenecks Using Program Slicing And Cognitive Complexity Metrics	Rilling, J.; Klemola, T.	2003	Program Comprehension, 2003. 11th IEEE International Workshop on	CE1	--
Image Quality Assessment Based On Local Orientation Distributions	Yue Wang; Tingting Jiang; Siwei Ma; Wen Gao	2010	Picture Coding Symposium (PCS), 2010	CE1	--
Impact Of Attribute Selection On Defect Proneness Prediction In Oo Software	Mishra, B.; Shukla, K.K.	2011	Computer and Communication Technology (ICCCT), 2011 2nd International Conference on	CE1	--
Impact Of Budget And Schedule Pressure On Software Development Cycle Time And Effort	Ning Nan; Harter, D.E.	2009	Software Engineering, IEEE Transactions on	CE1	--
Impact Of Uncertainty On The Prediction Of Airspace Complexity Of Congested Sectors	Sridhar, B.; Kulkarni, D.	2009	Digital Avionics Systems Conference, 2009. DASC '09. IEEE/AIAA 28th	CE1	--
Implementation Of Domcat: The Domain Complexity Analysis Tool For Natural Language Dialog Processing	Duvall S.	2007	Proceedings of the Richard Tapia Celebration of Diversity in Computing Conference 2007	CE1	--
Improved H.264 Rate Control By Enhanced Mad-Based Frame Complexity Prediction	Yi X., Ling N.	2006	Journal of Visual Communication and Image Representation	CE1	--
Improved Mb Mode Prediction In Extended Spatial Scalability With Error Resilient Coding	Shoaib, M.; Cai, A.	2010	Wireless Communications Networking and Mobile Computing (WiCOM), 2010 6th International Conference on	CE1	--
Improved Performance Models Of Web-Based Software Systems	Bogardi-Meszoly, Agnes; Levendovszky, Tihamer; Szeghegyi, Agnes	2009	Intelligent Engineering Systems, 2009. INES 2009. International Conference on	CE1	--
Improving Material Logistics Via Automation In An Existing Semiconductor Fab	Miller, D.; Menser, C.; Gustafson, B.	2004	Advanced Semiconductor Manufacturing, 2004. ASMC '04. IEEE Conference and Workshop	CE1	--
Improving Neural Network Promoter Prediction By Exploiting The Lengths Of Coding And Non-Coding Sequences	Caldwell R., Dai Y., Srivastava S., Lin Y.-X., Zhang R.	2008	Studies in Computational Intelligence	CE1	--
Improving Software Development Management Through Software Project Telemetry	Johnson, P.M.; Kou, H.; Paulding, M.; Zhang, Q.; Kagawa, A.; Yamashita, T.	2005	Software, IEEE	CE1	--
Improving Software Maintenance Size Metrics A Case Study: Automated Report Generation	Wirotyakun, Aphatsara; Netisopakul, Ponrudee	2012	Computer Science and Software Engineering (JCSSE), 2012 International Joint Conference on	CE1	--

System For Particle Monitoring In Hard Disk Drive Industry					
Indirectly Predicting The Maintenance Effort Of Open-Source Software	Yu L.	2006	Journal of Software Maintenance and Evolution	CE1	--
Influence Of Procedure Cloning On Wcet Prediction	Lokuciejewski, P.; Falk, H.; Schwarzer, M.; Marwedel, P.; Theiling, H.	2007	Hardware/Software Codesign and System Synthesis (CODES+ISSS), 2007 5th IEEE/ACM/IFIP International Conference on	CE1	--
Information Fusion In A Hybrid Tightly Coupled Gps/Dead-Reckoning Positioning System	Kacemi J., Reboul S., Benjelloun M.	2006	IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems	CE1	--
Information Theoretic Methods For Modeling Of Gene Regulatory Networks	Noor, A.; Serpedin, E.; Nounou, M.; Nounou, H.; Mohamed, N.; Chouchane, L.	2012	Computational Intelligence in Bioinformatics and Computational Biology (CIBCB), 2012 IEEE Symposium on	CE1	--
Integrating Dynamic Fab Capacity And Automation Models For 300 Mm Semiconductor Manufacturing	DeJong, C.D.; Fischbein, S.A.	2000	Simulation Conference, 2000. Proceedings. Winter	CE1	--
Integrating Open Source Software Into Software Engineering Curriculum: Challenges In Selecting Projects	Gokhale, S.S.; Smith, T.; McCartney, R.	2012	Software Engineering Education based on Real-World Experiences (EduRex), 2012 First International Workshop on	CE1	--
Integrating Testing With Reliability	Schneidewind N.	2009	Software Testing Verification and Reliability	CE1	--
Inter-Item Correlations Among Function Points	Kitchenham, B.; Kansala, K.	1993	Software Engineering, 1993. Proceedings., 15th International Conference on	CE1	--
Is Complexity Really The Enemy Of Software Security?	Shin Y., Williams L.	2008	Proceedings of the ACM Conference on Computer and Communications Security	CE1	--
Is Software Aging Related To Software Metrics?	Cotroneo, D.; Natella, R.; Pietrantuono, R.	2010	Software Aging and Rejuvenation (WoSAR), 2010 IEEE Second International Workshop on	CE1	--
Joint Rate-Distortion Model For H.264/Avc Rate Control	Zhou Y., Sun Y., Ahmad I., Sun S.	2009	Proceedings - International Conference on Image Processing, ICIP	CE1	--
Knowledge Based Quality-Driven Architecture Design And Evaluation	Ovaska E., Evesti A., Henttonen K., Palviainen M., Aho P.	2010	Information and Software Technology	CE1	--
Landscape Characterization Of Numerical Optimization Problems Using Biased Scattered Data	Munoz, Mario A.; Kirley, Michael; Halgamuge, Saman K.	2012	Evolutionary Computation (CEC), 2012 IEEE Congress on	CE1	--
Large Scale Analysis Of Web Revisitation Patterns	Adar E., Teevan J., Dumais S.T.	2008	Conference on Human Factors in Computing Systems - Proceedings	CE1	--
Link Prediction Based On Local Information	Yuxiao Dong; Qing Ke; Bai Wang; Bin Wu	2011	Advances in Social Networks Analysis and Mining (ASONAM), 2011 International Conference on	CE1	--
Machine Learning Based Call Admission Control Approaches: A Comparative Study	Bashar, A.; Parr, G.; McClean, S.; Scotney, B.; Nauck, D.	2010	Network and Service Management (CNSM), 2010 International Conference on	CE1	--
Material Analysis And Characterization Of Cesium Iodide (CsI) Coated C Fibers For Field Emission Applications	Vlahos, V.; Morgan, D.; Booske, J.H.	2008	Plasma Science, 2008. ICOPS 2008. IEEE 35th International Conference on	CE1	--
Measurement Based Investigations For Future Communication System Performance Evaluation	Pederen G.F., Nielsen J.o., Franek O., Andersen J.B., Pelosi M., Wang Y.	2009	Loughborough Antennas and Propagation Conference, LAPC 2009 - Conference Proceedings	CE1	--
Measurement Framework For Assessing Risks In Component-Based Software Development	Yi Ding; Napier, N.	2006	System Sciences, 2006. HICSS '06. Proceedings of the 39th Annual Hawaii International Conference on	CE1	--
Measuring Classification Complexity Of Image Databases: A Novel Approach	Rahman, A.F.R.; Fairhurst, M.C.	1999	Image Analysis and Processing, 1999. Proceedings. International Conference on	CE1	--
Measuring Complexity And Predictability In Networks With Multiscale Entropy Analysis	Riihijarvi, J.; Wellens, M.; Mahonen, P.	2009	INFOCOM 2009, IEEE	CE1	--
Measuring Design Quality By Measuring Design Complexity	Keating, M.	2000	Quality Electronic Design, 2000. ISQED 2000. Proceedings. IEEE 2000 First International Symposium on	CE1	--

Measuring Empirical Computational Complexity	Goldsmith S.F., Aiken A.S., Wilkerson D.S.	2007	6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, ESEC/FSE 2007	CE1	--
Measuring The Psychological Complexity Of Software Maintenance Tasks With The Halstead And Mccabe Metrics.	Curtis Bill, Sheppard Sylvia B., Milliman Phil, Borst M.A., Love Tom	1979	IEEE Transactions on Software Engineering	CE1	--
Metal Embedded Optical Fiber Sensors: Laser-Based Layered Manufacturing Procedures	Alemohammad H., Toyserkani E.	2011	Journal of Manufacturing Science and Engineering, Transactions of the ASME	CE1	--
Metrics And Methods For Benchmarking Of Rf Transmitter Behavioral Models With Application To The Development Of A Hybrid Memory Polynomial Model	Hammi, O.; Younes, M.; Ghannouchi, F.M.	2010	Broadcasting, IEEE Transactions on	CE1	--
Metrics Suite For Class Complexity	Michura, J.; Capretz, M.A.M.	2005	Information Technology: Coding and Computing, 2005. ITCC 2005. International Conference on	CE1	--
Mimo Radar Resource Allocation Using Posterior Cram?r-Rao Lower Bounds	Glass, J.D.; Smith, L.D.	2011	Aerospace Conference, 2011 IEEE	CE1	--
Mining Co-Location Relationships Among Bug Reports To Localize Fault-Prone Modules	Chen I.-X., Li C.-H., Yang C.-Z.	2010	IEICE Transactions on Information and Systems	CE1	--
Mining Developers' Communication To Assess Software Quality: Promises, Challenges, Perils	Di Penta M.	2012	2012 3rd International Workshop on Emerging Trends in Software Metrics, WETSoM 2012 - Proceedings	CE1	--
Mobility Prediction Based On Machine Learning	Anagnostopoulos, T.; Anagnostopoulos, C.; Hadjiefthymiades, S.	2011	Mobile Data Management (MDM), 2011 12th IEEE International Conference on	CE1	--
Model Selection For K-Nearest Neighbors Regression Using Vc Bounds	Cherkassky, V.; Yunqian Ma; Jun Tang	2003	Neural Networks, 2003. Proceedings of the International Joint Conference on	CE1	--
Model-Based Test For Analog Integrated Circuits	Barford, L.; Tufillaro, N.; Jefferson, S.; Khoche, A.	2007	Instrumentation and Measurement Technology Conference Proceedings, 2007. IMTC 2007. IEEE	CE1	--
Modeling Change Requests Due To Faults In A Large-Scale Telecommunication System	Jung H.-W., Lim Y., Chung C.-S.	2004	Journal of Systems and Software	CE1	--
Modeling Maintenance-Strategies With Rainbow Nets	Johnson, A.M., Jr.; Schoenfelder, M.A.; Lebold, D.	1992	Reliability and Maintainability Symposium, 1992. Proceedings., Annual	CE1	--
Modeling The Relationship Between Source Code Complexity And Maintenance Difficulty	Lanning, D.L.; Khoshgoftaar, T.M.	1994	Computer	CE1	--
Multi-Core Aware Process Mapping And Its Impact On Communication Overhead Of Parallel Applications	Rodrigues, E.R.; Madruga, F.L.; Navaux, P.O.A.; Panetta, J.	2009	Computers and Communications, 2009. ISCC 2009. IEEE Symposium on	CE1	--
Multifractal Characterization And Fuzzy Classification Of Lightning Strike Maps	Faghfouri, A.; Kinsner, W.; Swatek, D.	2004	Fuzzy Information, 2004. Processing NAFIPS '04. IEEE Annual Meeting of the	CE1	--
Multifractal Spectra Of Lightning Strike Maps Using Entropy And Wavelet Analyses	Faghfouri, A.; Kinsner, W.; Swatek, D.	2004	Electrical and Computer Engineering, 2004. Canadian Conference on	CE1	--
Multiobjective Optimization Of Multiple Scale Visual Quality Processing	Engelke, U.; Zepernick, H.-J.	2008	Multimedia Signal Processing, 2008 IEEE 10th Workshop on	CE1	--
Multipath Signal Simulation In A Dynamic Aircraft Landing Environment	Weiss J.P., Axelrad P.	2007	20th International Technical Meeting of the Satellite Division of The Institute of Navigation 2007 ION GNSS 2007	CE1	--
Multiscale Complexity Analysis Of Heart Rate Dynamics In Heart Failure: Preliminary Findings From The Music Study	Costa, M.; Cygankiewicz, I.; Zareba, W.; de Luna, A.B.; Goldberger, A.L.; Lobodzinski, S.	2006	Computers in Cardiology, 2006	CE1	--

Mutable Protection Domains: Towards A Component-Based System For Dependable And Predictable Computing	Parmer, G.; West, R.	2007	Real-Time Systems Symposium, 2007. RTSS 2007. 28th IEEE International	CE1	--
Near-Lossless Image Compression Schemes Based On Weighted Finite Automata Encoding And Adaptive Context Modelling	Bao, P.; Xiaolin Wu	1997	Compression and Complexity of Sequences 1997. Proceedings	CE1	--
Networks, Multicore, And Systems Evolution - Facing The Timing Beast	Ernst, Rolf	2008	Emerging Technologies and Factory Automation, 2008. ETFA 2008. IEEE International Conference on	CE1	--
Neural Net Analysis Of The Propensity For Change In Large Software Systems	Morphet, S.B.; Fawcett, J.; Bolazar, K.; Gungor, M.	2006	Neural Networks, 2006. IJCNN '06. International Joint Conference on	CE1	--
New Rate-Complexity-Quantization Modeling And Efficient Rate Control For H.264/Avc	Yimin Zhou; Yu Sun; Zhidan Feng; Shixin Sun	2008	Multimedia and Expo, 2008 IEEE International Conference on	CE1	--
New Rate-Distortion Modeling And Efficient Rate Control For H.264/Avc Video Coding	Zhou Y., Sun Y., Feng Z., Sun S.	2009	Signal Processing: Image Communication	CE1	--
No-Reference Video Quality Assessment In The Compressed Domain	Xiangyu Lin; Hanjie Ma; Lei Luo; Yaowu Chen	2012	Consumer Electronics, IEEE Transactions on	CE1	--
Nonlinear Signal Classification In The Framework Of High-Dimensional Shape Analysis In Reconstructed State Space	Yang S.	2005	IEEE Transactions on Circuits and Systems II: Express Briefs	CE1	--
Novel Methods To Demarcate Urban House Submarket - Cluster Analysis With Spatially Varying Relationships Between House Value And Attributes	Yu, Danlin; Jingyuan Yin; Feiyue Ye	2011	Smart and Sustainable City (ICSSC 2011), IET International Conference on	CE1	--
Novel Rate Control Scheme For Intra Frame Video Coding With Exponential Rate-Distortion Model On H.264/Avc	Tian L., Zhou Y., Sun Y.	2012	Journal of Visual Communication and Image Representation	CE1	--
Novel Rate-Quantization Model-Based Rate Control With Adaptive Initialization For Spatial Scalable Video Coding	Sudeng Hu; Hanli Wang; Kwong, S.; Kuo, C.J.	2012	Industrial Electronics, IEEE Transactions on	CE1	--
Novel Spatio-Temporal Structural Information Based Video Quality Metric	Yue Wang; Tingting Jiang; Siwei Ma; Wen Gao	2012	Circuits and Systems for Video Technology, IEEE Transactions on	CE1	--
Object Oriented Metrics Useful In The Prediction Of Class Testing Complexity	Bluemke, I.	2001	Euromicro Conference, 2001. Proceedings. 27th	CE1	--
Objective Video Quality Assessment Methods: A Classification, Review, And Performance Comparison	Chikkerur, S.; Sundaram, V.; Reisslein, M.; Karam, L.J.	2011	Broadcasting, IEEE Transactions on	CE1	--
On Feature Selection For Genomic Signal Processing And Data Mining	Kung, S.Y.	2007	Machine Learning for Signal Processing, 2007 IEEE Workshop on	CE1	--
On The Effect Of The Query In Ir-Based Concept Location	Haiduc, S.; Marcus, A.	2011	Program Comprehension (ICPC), 2011 IEEE 19th International Conference on	CE1	--
On The Foundations Of System Identification	Caines, P.E.	1988	Decision and Control, 1988., Proceedings of the 27th IEEE Conference on	CE1	--
On The Impact Of Software Product Dissimilarity On Software Quality Models	Khoshgoftaar, T.M.; Lanning, D.L.	1994	Software Reliability Engineering, 1994. Proceedings., 5th International Symposium on	CE1	--
On The Jensen?Shannon Divergence And Variational Distance	Tsai, S.-C.; Tzeng, W.-G.; Wu, H.-L.	2005	Information Theory, IEEE Transactions on	CE1	--
On The Relation Between External Software Quality And Static Code Analysis	Plosch, R.; Gruber, H.; Hentschel, A.; Pomberger, G.; Schiffer, S.	2008	Software Engineering Workshop, 2008. SEW '08. 32nd Annual IEEE	CE1	--
On The Relation Between Rhythm Complexity Measures And Human	Thul E., Toussaint G.T.	2008	ACM International Conference Proceeding Series	CE1	--

Rhythmic Performance					
On The Use Of Color Appearance Modeling For Efficient Compressed-Domain Image Enhancement	Chatzigiorgaki, M.; Skodras, A.N.	2009	Image Processing (ICIP), 2009 16th IEEE International Conference on	CE1	--
On The Value Of Learning From Defect Dense Components For Software Defect Prediction	Zhang H., Nelson A., Menzies T.	2010	ACM International Conference Proceeding Series	CE1	--
On-Line Estimation Of Internet Path Performance: An Application Perspective	Tao, S.; Guerin, R.	2004	INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies	CE1	--
Optimal Bit Allocation And Efficient Rate Control For H.264/Avc Based On General Rate-Distortion Model And Enhanced Coding Complexity Measure	Xie Z., Bao Z., Xu C., Zhang G.	2010	IET Image Processing	CE1	--
Optimization Of Hierarchical 3drs Motion Estimators For Picture Rate Conversion	Heinrich, A.; Bartels, C.; van der Vleuten, R.J.; Cordes, C.N.; de Haan, G.	2011	Selected Topics in Signal Processing, IEEE Journal of	CE1	--
Optimum Structural Design Of Robotic Manipulators With Fiber Reinforced Composite Materials	Saravanos D.A., Lamancusa J.S.	1990	Computers and Structures	CE1	--
Optimum Two-Dimensional Uniform Spatial Sampling For Microwave Sar-Based Nde Imaging Systems	Case, J.T.; Ghasr, M.T.; Zoughi, R.	2011	Instrumentation and Measurement, IEEE Transactions on	CE1	--
Order Of Nonlinearity As A Complexity Measure For Models Generated By Symbolic Regression Via Pareto Genetic Programming	Vladislavleva, E.J.; Smits, G.F.; den Hertog, D.	2009	Evolutionary Computation, IEEE Transactions on	CE1	--
Oxidation Of Si Nanocrystals Fabricated By Ultralow-Energy Ion Implantation In Thin <equation><font Face='Verdana'>Si</Font><Font Face='Verdana'>O</Font><Sub>2< /Sub></Equation> Layers	Coffin, H.; Bonafos, C.; Schamm, S.; Cherkashin, N.; Assayag, G. Ben; Claverie, A.; Respaud, M.; Dimitrakis, P.; Normand, P.	2006	Journal of Applied Physics	CE1	--
Parallel Software Engineering - Goals 2000	Murphy, C.	1995	Computer Software and Applications Conference, 1995. COMPSAC 95. Proceedings., Nineteenth Annual International	CE1	--
Parametric Modelling Of Temporal Variations In Radon Concentrations In Homes	Revzan, K.L.; Turk, B.H.; Nero, A.V.; Sextro, R.G.	1988	Nuclear Science, IEEE Transactions on	CE1	--
Pareto Optimalweighting Of Structural Impairments For Wireless Imaging Quality Assessment	Engelke, U.; Zepernick, H.-J.	2008	Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on	CE1	--
Pattern Spectrum And Multiscale Shape Representation	Maragos, P.	1989	Pattern Analysis and Machine Intelligence, IEEE Transactions on	CE1	--
Performance By Design [railway Industry]	Halliday, B.	2004	On the Right Lines - Systems Engineering for the Railway Industry, IEE Seminar	CE1	--
Performance Measurement, Visualization And Modeling Of Parallel And Distributed Programs Using The Aims Toolkit	Yan Jerry, Sarukkai Sekhar, Mehra Pankaj	1995	Software - Practice and Experience	CE1	--
Performance Optimization Of Throttled Time-Warp Simulation	Seng Chuan Tay; Yong Meng Teo	2001	Simulation Symposium, 2001. Proceedings. 34th Annual	CE1	--
Performance Predictable Design Of Robust Motion Controllers For High-Precision Servo Systems	Bong Keun Kim; Wan Kyun Chung	2001	American Control Conference, 2001. Proceedings of the 2001	CE1	--
Performance Tuning Of Sliding Mode Controllers: Structural Analysis Approach	Bong Keun Kim; Wan Kyun Chung; Il Hong Suh	2001	American Control Conference, 2001. Proceedings of the 2001	CE1	--
Pilot Perceptions Of Airspace Complexity. Part 2	Riley, V.; Chatterji, G.; Johnson, W.; Mogford, R.; Kopardekar, P.; Sieira, E.; Landing, M.; Lawton, G.	2004	Digital Avionics Systems Conference, 2004. DASC 04. The 23rd	CE1	--

Point/Counterpoint	Weller, Ed; Card, David; Curtis, Bill; Raczynski, Bob	2008	Software, IEEE	CE1	--
Power Supply Noise In Socs: Metrics, Management, And Measurement	Arabi, K.; Saleh, R.; Meng Xiongfei	2007	Design & Test of Computers, IEEE	CE1	--
Pre-Layout Physical Connectivity Prediction With Application In Clustering-Based Placement	Qinghua Liu; Marek-Sadowska, M.	2005	Computer Design: VLSI in Computers and Processors, 2005. ICCD 2005. Proceedings. 2005 IEEE International Conference on	CE1	--
Pre-Layout Wire Length And Congestion Estimation	Liu Q., Marek-Sadowska M.	2004	Proceedings - Design Automation Conference	CE1	--
Predict The Neurological Recovery Under Hypothermia After Cardiac Arrest Using C0 Complexity Measure Of Eeg Signals	Yueli Lu,; Dineng Jiang,; Xiaofeng Jia,; Yihong Qiu,; Yisheng Zhu,; Nitish Thakor,; Shanbao Tong,	2008	Engineering in Medicine and Biology Society, 2008. EMBS 2008. 30th Annual International Conference of the IEEE	CE1	--
Predicting Change Propagation In Software Systems	Hassan, A.E.; Holt, R.C.	2004	Software Maintenance, 2004. Proceedings. 20th IEEE International Conference on	CE1	--
Predicting Faults Using The Complexity Of Code Changes	Hassan, A.E.	2009	Software Engineering, 2009. ICSE 2009. IEEE 31st International Conference on	CE1	--
Predicting Function Changes By Mining Revision History	Malik, H.; Shakshuki, E.	2010	Information Technology: New Generations (ITNG), 2010 Seventh International Conference on	CE1	--
Predicting G-Protein-Coupled Receptor Classes Based On Adaptive K-Nearest Neighbor Algorithm	Xuan Xiao; Wang-ren Qiu	2010	Control and Decision Conference (CCDC), 2010 Chinese	CE1	--
Predicting Labor Cost Through It Management Complexity Metrics	Yixin Diao; Keller, A.; Parekh, S.; Marinov, V.V.	2007	Integrated Network Management, 2007. IM '07. 10th IFIP/IEEE International Symposium on	CE1	--
Predicting Noise Filtering Efficacy With Data Complexity Measures For Nearest Neighbor Classification	Saez J.A., Luengo J., Herrera F.	2012	Pattern Recognition	CE1	--
Predicting Osseous Changes In Ankle Fractures	Webber, R.L.; Underhill, T.E.; Horton, R.A.; Dixon, R.L.; Pope, T.L., Jr.	1993	Engineering in Medicine and Biology Magazine, IEEE	CE1	--
Predicting Player Behavior In Tomb Raider: Underworld	Mahlmann, T.; Drachen, A.; Togelius, J.; Canossa, A.; Yannakakis, G.N.	2010	Computational Intelligence and Games (CIG), 2010 IEEE Symposium on	CE1	--
Predicting Software Suitability Using A Bayesian Belief Network	Beaver, J.M.; Schiavone, G.A.; Berrios, J.S.	2005	Machine Learning and Applications, 2005. Proceedings. Fourth International Conference on	CE1	--
Prediction Of Area And Length Complexity Measures For Binary Decision Diagrams	Beg A., Chandana Prasad P.W.	2010	Expert Systems with Applications	CE1	--
Prediction Of Fault-Prone Software Modules Using A Generic Text Discriminator	Mizuno O., Kikuno T.	2008	IEICE Transactions on Information and Systems	CE1	--
Prediction Of Protein Catalytic Residues By Local Structural Rigidity	Yu-Tung Chien; Shao-Wei Huang	2012	Complex, Intelligent and Software Intensive Systems (CISIS), 2012 Sixth International Conference on	CE1	--
Preemptive Virtual Clock: A Flexible, Efficient, And Cost-Effective Qos Scheme For Networks-On-Chip	Grot, B.; Keckler, S.W.; Mutlu, O.	2009	Microarchitecture, 2009. MICRO-42. 42nd Annual IEEE/ACM International Symposium on	CE1	--
Pro-Active Page Replacement For Scientific Applications: A Characterization	Murali Vilayannur; Anand Sivasubramaniam; Kandemir, M.	2005	Performance Analysis of Systems and Software, 2005. ISPASS 2005. IEEE International Symposium on	CE1	--
Problem Characterization In Tracking/Fusion Algorithm Evaluation	Chee-Yee Chong	2001	Aerospace and Electronic Systems Magazine, IEEE	CE1	--
Production Evaluation Of Automated Reticle Defect Printability Prediction Application	Howard, William B.; Pomeroy, Scott; Moses, Raphael; Thaler, Thomas	2007	Mask and Lithography Conference (EMLC), 2007 23rd European	CE1	--
Programmer-Based Fault Prediction	Ostrand T.J., Weyuker E.J., Bell R.M.	2010	ACM International Conference Proceeding Series	CE1	--
Progress And Quality Modeling Of Requirements Analysis Based On	Junwei Ge; Yiqiu Fang	2009	Software Engineering, 2009. WCSE '09. WRI World Congress on	CE1	--

Chaos					
Putting Usable Intelligence Into Multimedia Applications	Maybury, M.T.	1999	Multimedia Computing and Systems, 1999. IEEE International Conference on	CE1	--
Quality Assessment Of Video Content For Hd Iptv Applications	Wei Li; Issa, O.; Hong Liu; Speranza, F.; Renaud, R.	2009	Multimedia, 2009. ISM '09. 11th IEEE International Symposium on	CE1	--
Quality Of Code Can Be Planned And Automatically Controlled	Bugayenko, Y.	2009	Advances in System Testing and Validation Lifecycle, 2009. VALID '09. First International Conference on	CE1	--
Quantifying Ruggedness Of Continuous Landscapes Using Entropy	Malan, K.M.; Engelbrecht, A.P.	2009	Evolutionary Computation, 2009. CEC '09. IEEE Congress on	CE1	--
Quantifying The Effect Of Learning On Recurrent Spiking Neurons	Brodu N.	2007	IEEE International Conference on Neural Networks - Conference Proceedings	CE1	--
Quantitative Analysis Of Faults And Failures In A Complex Software System	Fenton, N.E.; Ohlsson, N.	2000	Software Engineering, IEEE Transactions on	CE1	--
Quantitatively Measuring Object-Oriented Couplings	Offutt J., Abdurazik A., Schach S.R.	2008	Software Quality Journal	CE1	--
Query-Time Entity Resolution	Bhattacharya I., Getoor L.	2007	Journal of Artificial Intelligence Research	CE1	--
Ranking Refactoring Suggestions Based On Historical Volatility	Tsantalis, N.; Chatzigeorgiou, A.	2011	Software Maintenance and Reengineering (CSMR), 2011 15th European Conference on	CE1	--
Rapid Dissemination Of Light Transport Models On The Web	Baranoski, G.V.G.; Dimson, T.; Chen, T.F.; Kimmel, B.; Yim, D.; Miranda, E.	2012	Computer Graphics and Applications, IEEE	CE1	--
Rate Control For H.264/Avc Using Enhanced Coding Complexity Measure	Zhengguang Xie; Zhihua Bao; Chen Xu; Guoan Zhang; Shibin Zhang; Yongjie Yang	2008	Signal Processing, 2008. ICSP 2008. 9th International Conference on	CE1	--
Rate Control Using Enhanced Frame Complexity Measure For H.264 Video	Xiaoquan Yi; Nam Ling	2004	Signal Processing Systems, 2004. SIPS 2004. IEEE Workshop on	CE1	--
Rate Controlling Based On Effective Source-Complexity Metrics Appropriated To H.264/Avc	Din-Yuen Chan; Zih-Siang Lin; Pei-Shan Wu	2010	Computer Symposium (ICS), 2010 International	CE1	--
Rate-Distortion Speech Coding With A Minimum Discrimination Information Distortion Measure.	Gray Robert M., Gray Jr. Augustine H., Robolledo Guillermo, Shore John E.	1981	IEEE Transactions on Information Theory	CE1	--
Real Time Prediction Of Earthquake Ground Motions And Structural Responses By Statistic And Fuzzy Logic	Kawamura, H.; Tani, A.; Yamada, M.; Tsunoda, K.	1990	Uncertainty Modeling and Analysis, 1990. Proceedings., First International Symposium on	CE1	--
Real-Time Estimation Of Depth Of Anaesthesia Using The Mutual Information Of Electroencephalograms	Liyu Huang; Fengchi Ju; Enke Zhang; Jingzhi Cheng	2003	Neural Engineering, 2003. Conference Proceedings. First International IEEE EMBS Conference on	CE1	--
Reducing Features To Improve Code Change Based Bug Prediction	Shivaji, S.; Whitehead, Jr., E.; Akella, R.; Kim, S.	2012	Software Engineering, IEEE Transactions on	CE1	--
Reducing Message Overhead Of Aodv Routing Protocol In Urban Area By Using Link Availability Prediction	Ghanbarzadeh, R.; Meybodi, M.R.	2010	Computer Research and Development, 2010 Second International Conference on	CE1	--
Reducing Wasted Development Time Via Continuous Testing	Saff, D.; Ernst, M.D.	2003	Software Reliability Engineering, 2003. ISSRE 2003. 14th International Symposium on	CE1	--
Reduction Of Wind Induced Antenna Pointing Errors Using Structural Analysis And Testing	Brannan, P.; Black, I.; Crewe, A.	1991	Antennas and Propagation, 1991. ICAP 91., Seventh International Conference on (IEE)	CE1	--
Refined Methods For Creating Realistic Haptic Virtual Textures From Tool-Mediated Contact Acceleration Data	Culbertson, H.; Romano, J.M.; Castillo, P.; Mintz, M.; Kuchenbecker, K.J.	2012	Haptics Symposium (HAPTICS), 2012 IEEE	CE1	--
Reflect And Correct: A Misclassification Prediction Approach To Active Inference	Bilgic M., Getoor L.	2009	ACM Transactions on Knowledge Discovery from Data	CE1	--

Relating Static And Dynamic Machine Code Measurements	Davidson Jack W., Rabung John R., Whalley David B.	1992	IEEE Transactions on Computers	CE1	--
Relational Ibl In Music With A New Structural Similarity Measure	Tobudic A., Widmer G.	2003	Lecture Notes in Artificial Intelligence (Subseries of Lecture Notes in Computer Science)	CE1	--
Relaxation Behavior Of Undoped In<Inf>X</Inf>Ga<Inf>1-X</Inf>P 0.5?X?0.7 Grown On Gaas By Atomic Layer Molecular-Beam Epitaxy	Gonzalez, L.; Gonzalez, Y.; Aragon, G.; Castro, M. J.; Dotor, M. L.; Dunstan, D. J.	1996	Journal of Applied Physics	CE1	--
Reliability, Availability And Maintainability Modelling - Metrics And Models	Best, N.	2004	Railway System Modelling - Not Just for Fun, 2004. The IEE Seminar on	CE1	--
Resource Availability Evaluation In Service Grid Environment	Hu Zhoujun; Hu Zhigang; Liu Zhenhua	2007	Asia-Pacific Service Computing Conference, The 2nd IEEE	CE1	--
Rethinking The Recommender Research Ecosystem: Reproducibility, Openness, And Lenskit	Ekstrand M.D., Ludwig M., Konstan J.A., Riedl J.T.	2011	RecSys'11 - Proceedings of the 5th ACM Conference on Recommender Systems	CE1	--
Reusability Hypothesis Verification Using Machine Learning Techniques: A Case Study	Yida Mao; Sahraoui, H.A.; Lounis, H.	1998	Automated Software Engineering, 1998. Proceedings. 13th IEEE International Conference on	CE1	--
Revisiting Measurement Of Software Complexity	Wohlin, C.	1996	Software Engineering Conference, 1996. Proceedings. 1996 Asia-Pacific	CE1	--
Risk Prediction In Erp Projects: Classification Of Reengineered Business Processes	Camara, M.S.; Kermad, L.; El Mhamedi, A.	2006	Computational Intelligence for Modelling, Control and Automation, 2006 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, International Conference on	CE1	--
Semi-Individual Wire-Length Prediction With Application To Logic Synthesis	Qinghua Liu; Marek-Sadowska, M.	2006	Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on	CE1	--
Short Term Link Performance Modeling For Ml Receivers With Mutual Information Per Bit Metrics	Sayana, K.; Zhuang, J.; Stewart, K.	2008	Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE	CE1	--
Short-Term Prediction Of Railway Passenger Flow Based On Rbf Neural Network	Yuanchun Huang; Haize Pan	2011	Computational Sciences and Optimization (CSO), 2011 Fourth International Joint Conference on	CE1	--
Simplified Dynamic Density Based Capacity Estimation	Chok Fung Lai; Zelinski, S.	2009	Digital Avionics Systems Conference, 2009. DASC '09. IEEE/AIAA 28th	CE1	--
Sinusoidal Frequency Estimation In Chaotic Noise	Leung, H.; Xinping Huang	1995	Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on	CE1	--
Slice-Based Dynamic Memory Modelling - A Case Study	Sivagurunathan, Y.; Harman, M.; Sivagurnathan, B.	2002	Computer Software and Applications Conference, 2002. COMPSAC 2002. Proceedings. 26th Annual International	CE1	--
Sofas: A Lightweight Architecture For Software Analysis As A Service	Ghezzi, G.; Gall, H.C.	2011	Software Architecture (WICSA), 2011 9th Working IEEE/IFIP Conference on	CE1	--
Software Complexity Level Determination Using Software Effort Estimation Use Case Points Metrics	Yavari, Y.; Afsharchi, M.; Karami, M.	2011	Software Engineering (MySEC), 2011 5th Malaysian Conference in	CE1	--
Software Forensics For Discriminating Between Program Authors Using Case-Based Reasoning, Feedforward Neural Networks And Multiple Discriminant Analysis	Macdonell, S.G.; Gray, A.R.; MacLennan, G.; Sallis, P.J.	1999	Neural Information Processing, 1999. Proceedings. ICONIP '99. 6th International Conference on	CE1	--
Software Metric For Logical Errors And Integration Testing Effort	Leach Ronald J., Coleman Don M.	1997	COMPASS - Proceedings of the Annual Conference on Computer Assurance	CE1	--
Software Product Metrics	Li, W.	1999	Potentials, IEEE	CE1	--
Software Project Schedule Variance Prediction Using Bayesian Network	Xiaoxu Wang; Chaoying Wu; Lin Ma	2010	Advanced Management Science (ICAMS), 2010 IEEE International Conference on	CE1	--
Software Quality Metrics In Space	Valette, V.; Vallee, F.	1992	Software Reliability Engineering,	CE1	--

Systems			1992. Proceedings., Third International Symposium on		
Software Quality Model And Framework With Applications In Industrial Context	Schrettner, L.; Jeno Fu?lo?p, L.; Besze?des, A.; Kiss, A.; Gyimo?thy, T.	2012	Software Maintenance and Reengineering (CSMR), 2012 16th European Conference on	CE1	--
Software Reliability, Availability, And Maintainability Engineering System (SOFT-RAMES)	Edson, B.; Hansen, B.; Larter, P.	1996	Reliability and Maintainability Symposium, 1996 Proceedings. 'International Symposium on Product Quality and Integrity'., Annual	CE1	--
Some Complexity Metrics For Object-Oriented Programs Based On Information Flow: A Study Of C++ Programs	Lee Yen-Sung, Liang Bin-Shiang, Wang Feng-Jian	1994	Journal of Information Science and Engineering	CE1	--
Source Code Comprehension Strategies And Metrics To Predict Comprehension Effort In Software Maintenance And Evolution Tasks - An Empirical Study With Industry Practitioners	Nishizono, K.; Morisaki, S.; Vivanco, R.; Matsumoto, K.	2011	Software Maintenance (ICSM), 2011 27th IEEE International Conference on	CE1	--
Space-Time Duality In Digital Filter Structures.	Fam Adly T.	1983	IEEE Transactions on Acoustics, Speech, and Signal Processing	CE1	--
Spectroscopic Ellipsometry Studies Of Very Thin Thermally Grown Sio<Inf>2</Inf> Films: Influence Of Oxidation Procedure On Oxide Quality And Stress	Boultadakis, S.; Logothetidis, S.; Papadopoulos, A.; Vouroutzis, N.; Zorba, Ph.; Girginoudi, D.; Thanailakis, A.	1995	Journal of Applied Physics	CE1	--
Speech Denoising Based On Perceptual Weighting Filter	Dong E., Pu X.	2008	International Conference on Signal Processing Proceedings, ICSP	CE1	--
Stability And Scalability In Global Routing	Sung Kyu Han; Kwangok Jeong; Kahng, A.B.; Jingwei Lu	2011	System Level Interconnect Prediction (SLIP), 2011 13th International Workshop on	CE1	--
Statistical Data Reduction For Efficient Application Performance Monitoring	Lingyun Yang; Schopf, J.M.; Dumitrescu, C.L.; Foster, I.	2006	Cluster Computing and the Grid, 2006. CCGRID 06. Sixth IEEE International Symposium on	CE1	--
Statistical Power And Performance Modeling For Optimizing The Energy Efficiency Of Scientific Computing	Subramaniam, B.; Wu-chun Feng	2010	Green Computing and Communications (GreenCom), 2010 IEEE/ACM Int'l Conference on & Int'l Conference on Cyber, Physical and Social Computing (CPSCom)	CE1	--
Stochastic Complexity Measures For Physiological Signal Analysis	Rezek, I.A.; Roberts, S.J.	1998	Biomedical Engineering, IEEE Transactions on	CE1	--
Strain Dependence Of Dielectric Properties And Reliability Of High-K Thin Films	Suzuki, K.; Imasaki, K.; Ito, Y.; Miura, H.	2009	Simulation of Semiconductor Processes and Devices, 2009. SISPAD '09. International Conference on	CE1	--
Structural And Dielectric Characteristics Of 0.50 Bi₂O ₃- 0.25 V₂O₅- 0.25 Srb₄O₇ Glass-Ceramic	Shankar, M.V.; Varma, K.B.R.	1996	Applications of Ferroelectrics, 1996. ISAF '96., Proceedings of the Tenth IEEE International Symposium on	CE1	--
Structural And Dielectric Characteristics Of Bi₂O₃- TiO₂- SrB₄O₇ Glasses	Varma, K.B.R.; Shankar, M.V.	1996	Applications of Ferroelectrics, 1996. ISAF '96., Proceedings of the Tenth IEEE International Symposium on	CE1	--
Structural Characteristics Of Gene Networks For Colon Cancer	Shudong Wang; Kaikai Li; Xiuzhu Xu; Dazhi Meng; Dashun Xu	2011	Signal Processing, Communications and Computing (ICSPCC), 2011 IEEE International Conference on	CE1	--
Structural Damage Detection By Integrating Independent Component Analysis And Artificial Neural Networks	Song H., Zhong L., Moon F.	2005	Proceedings of the 2005 International Conference on Machine Learning; Models, Technologies and Applications, MLMTA'05	CE1	--
Structural Identification: The Computation Of The Generic Mcmillan Degree	Karcanias, N.; Sagianos, E.; Milonidis, E.	2005	Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on	CE1	--

Structural Metrics Of High-Temperature Lattice Conductivity	Huang, B. L.; Kaviany, M.	2006	Journal of Applied Physics	CE1	--
Structural Predictors Of Tie Formation In Twitter: Transitivity And Mutuality	Golder, S.A.; Yardi, S.	2010	Social Computing (SocialCom), 2010 IEEE Second International Conference on	CE1	--
Structuring Cognitive Information For Software Complexity Measurement	Auprasert, B.; Limpiyakorn, Y.	2009	Computer Science and Information Engineering, 2009 WRI World Congress on	CE1	--
Study Of Strip-Filling For Old Room And Pillar Mining Goaf	Liu Cong-liang; Tan Zhi-xiang; Li Pei-xian; Deng Ka-zhong	2011	Electric Technology and Civil Engineering (ICETCE), 2011 International Conference on	CE1	--
Studying The Impact Of Social Interactions On Software Quality	Bettenburg N., Hassan A.E.	2012	Empirical Software Engineering	CE1	--
Studying The Impact Of Social Structures On Software Quality	Bettenburg, N.; Hassan, A.E.	2010	Program Comprehension (ICPC), 2010 IEEE 18th International Conference on	CE1	--
Symbolic Performance Estimation Of Speculative Parallel Programs	Gautama H., Van Gemund A.J.C.	2003	Parallel Processing Letters	CE1	--
System Design Metrics: A Review And Perspective	Ince, D.C.; Sheppard, M.J.	1988	Software Engineering, 1988 Software Engineering 88., Second IEE/BCS Conference:	CE1	--
Task Assignment With Cache Partitioning And Locking For Wcet Minimization On Mpsoc	Tiantian Liu; Yingchao Zhao; Minming Li; Xue, C.J.	2010	Parallel Processing (ICPP), 2010 39th International Conference on	CE1	--
Texture Segmentation By Multiscale Aggregation Of Filter Responses And Shape Elements	Galun M., Sharon E., Basri R., Brandt A.	2003	Proceedings of the IEEE International Conference on Computer Vision	CE1	--
The Conceptual Cohesion Of Classes	Marcus, A.; Poshyvanyk, D.	2005	Software Maintenance, 2005. ICSM'05. Proceedings of the 21st IEEE International Conference on	CE1	--
The Conceptual Coupling Metrics For Object-Oriented Systems	Poshyvanyk, D.; Marcus, A.	2006	Software Maintenance, 2006. ICSM '06. 22nd IEEE International Conference on	CE1	--
The Design Of Joint Source And Channel Trellis Waveform Coders	Ayanoglu, E.; Gray, R.	1987	Information Theory, IEEE Transactions on	CE1	--
The Detection Of Fault-Prone Programs	Munson, J.C.; Khoshgoftaar, T.M.	1992	Software Engineering, IEEE Transactions on	CE1	--
The Detection Of Seizure Vulnerable Period From Epidural Eeg Recordings Of Epilepsy Rat	Wenyan Jia; Xuan Liu; Xiaorong Gao; Liang Guo; Yang Lei; Yixuan Ku; Shangkai Gao	2005	Neural Engineering, 2005. Conference Proceedings. 2nd International IEEE EMBS Conference on	CE1	--
The Evolution Of Source Folder Structure In Actively Evolved Open Source Systems	Capiluppi, A.; Morisio, M.; Ramil, J.F.	2004	Software Metrics, 2004. Proceedings. 10th International Symposium on	CE1	--
The Fast Methodology For High-Speed Soc/Computer Simulation	Chiou, D.; Dam Sunwoo; Joonsoo Kim; Patil, N.; Reinhart, W.H.; Johnson, D.E.; Zheng Xu	2007	Computer-Aided Design, 2007. ICCAD 2007. IEEE/ACM International Conference on	CE1	--
The Impact Of Preprocessing On Forecasting Electrical Load: An Empirical Evaluation Of Segmenting Time Series Into Subseries	Crone, S.F.; Kourentzes, N.	2011	Neural Networks (IJCNN), The 2011 International Joint Conference on	CE1	--
The Influence Of Organizational Structure On Software Quality	Nagappan, N.; Murphy, B.; Basili, V.	2008	Software Engineering, 2008. ICSE '08. ACM/IEEE 30th International Conference on	CE1	--
The Limited Impact Of Individual Developer Data On Software Defect Prediction	Bell R.M., Ostrand T.J., Weyuker E.J.	2011	Empirical Software Engineering	CE1	--
The Mechanism Of Leader Breakdown In Electronegative Gases	Niemeyer, L.; Ullrich, L.; Wiegart, N.	1989	Electrical Insulation, IEEE Transactions on	CE1	--
The Misuse Of The Nasa Metrics Data Program Data Sets For Automated Software Defect Prediction	Gray, David; Bowes, David; Davey, Neil; Sun, Yi; Christianson, Bruce	2011	Evaluation & Assessment in Software Engineering (EASE 2011), 15th Annual Conference on	CE1	--
The Organization Of Interaction Design Pattern Languages Alongside The Design Process	Hubscher C., Pauwels S.L., Roth S.P., Bargas-Avila J.A., Opwis K.	2011	Interacting with Computers	CE1	--

The Road Not Taken: Estimating Path Execution Frequency Statically	Buse, R.P.L.; Weimer, W.	2009	Software Engineering, 2009. ICSE 2009. IEEE 31st International Conference on	CE1	--
The Role Of The Measure Of Functional Complexity In Effort Estimation	Lavazza L., Robiolo G.	2010	ACM International Conference Proceeding Series	CE1	--
The Spatiotemporal Change Of Urban Form In Nanjing, China: Based On Sleuth And Spatial Metrics Analysis	Zhenlong Zhang; Lingde Jiang; Rui Peng; Yixing Yin	2010	Geoinformatics, 2010 18th International Conference on	CE1	--
Theoretical And Experimental Analysis Of A Cycloidal Speed Reducer	Gorla C., Davoli P., Rosa F., Longoni C., Chiozzi F., Samarani A.	2008	Journal of Mechanical Design, Transactions of the ASME	CE1	--
Thermal Characterization Of Dbc And Mmc Stacks For Power Modules	Fusaro, J.M.; Romero, G.L.; Rodriguez, P.; Martinez, J.L., Jr.	1996	Industry Applications Conference, 1996. Thirty-First IAS Annual Meeting, IAS '96., Conference Record of the 1996 IEEE	CE1	--
Thresholds For Error Probability Measures Of Business Process Models	Mendling J., Sanchez-Gonzalez L., Garcia F., La Rosa M.	2012	Journal of Systems and Software	CE1	--
Thresholds For Object-Oriented Measures	Benlarbi, S.; El Emam, K.; Goel, N.; Rai, S.	2000	Software Reliability Engineering, 2000. ISSRE 2000. Proceedings. 11th International Symposium on	CE1	--
To Use Or Not To Use? The Metrics To Measure Software Quality (Developers' View)	Gyimothy, Tibor	2009	Software Maintenance and Reengineering, 2009. CSMR '09. 13th European Conference on	CE1	--
Topic-Based Defect Prediction: Nier Track	Tung Thanh Nguyen; Nguyen, T.N.; Tu Minh Phuong	2011	Software Engineering (ICSE), 2011 33rd International Conference on	CE1	--
Totalprof: A Fast And Accurate Retargetable Source Code Profiler	Gao L., Huang J., Ceng J., Leupers R., Ascheid G., Meyr H.	2009	Embedded Systems Week 2009 - 7th IEEE/ACM International Conference on Hardware/Software-Co-Design and System Synthesis, CODES+ISSS 2009	CE1	--
Toward A Measure Of Classification Complexity In Gene Expression Signatures	Kamath, Vidya; Yeatman, Timothy J.; Eschrich, Steven A.	2008	Engineering in Medicine and Biology Society, 2008. EMBS 2008. 30th Annual International Conference of the IEEE	CE1	--
Toward Air Traffic Complexity Assessment In New Generation Air Traffic Management Systems	Prandini, M.; Piroddi, L.; Puechmorel, S.; Brazdilova, S.L.	2011	Intelligent Transportation Systems, IEEE Transactions on	CE1	--
Toward The Software Realization Of A Gsm Base Station	Turletti, T.; Bentzen, H.J.; Tennenhouse, D.	1999	Selected Areas in Communications, IEEE Journal on	CE1	--
Toward The Use Of Automated Static Analysis Alerts For Early Identification Of Vulnerability- And Attack-Prone Components	Gegick, M.; Williams, L.	2007	Internet Monitoring and Protection, 2007. ICIMP 2007. Second International Conference on	CE1	--
Towards A Software Failure Cost Impact Model For The Customer: An Analysis Of An Open Source Product	Gitzel R., Krug S., Brhel M.	2010	ACM International Conference Proceeding Series	CE1	--
Towards Estimating Physical Properties Of Embedded Systems Using Software Quality Metrics	Corre?a, U.B.; Lamb, L.; Carro, L.; Brisolara, L.; Mattos, J.	2010	Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on	CE1	--
Towards Integrating Water Prediction And Control Technology	van Overloop, P.J.; Negenborn, R.R.; Schwanenberg, D.; De Schutter, B.	2011	Networking, Sensing and Control (ICNSC), 2011 IEEE International Conference on	CE1	--
Tracking Hardware Evolution	Nacif, J.A.; Silva, T.S.F.; Vieira, L.F.M.; Vieira, A.B.; Fernandes, A.O.; Coelho, C.	2011	Quality Electronic Design (ISQED), 2011 12th International Symposium on	CE1	--
Trajectory-Based Complexity (TBX): A Modified Aircraft Count To Predict Sector Complexity During Trajectory-Based Operations	Prevot, Thomas; Lee, Paul. U.	2011	Digital Avionics Systems Conference (DASC), 2011 IEEE/AIAA 30th	CE1	--
Understanding The Nexus Of Terrorist Web Sites	Xu J., Chen H.	2008	Studies in Computational Intelligence	CE1	--
Understanding Website Complexity: Measurements,	Butkiewicz M., Madhyastha H.V., Sekar V.	2011	Proceedings of the ACM SIGCOMM Internet Measurement Conference,	CE1	--

Metrics, And Implications			IMC		
Use Of A Genetic Algorithm To Identify Source Code Metrics Which Improves Cognitive Complexity Predictive Models	Vivanco, R.	2007	Program Comprehension, 2007. ICPC '07. 15th IEEE International Conference on	CE1	--
Using A Combination Of Measurement Tools To Extract Metrics From Open Source Projects	Bakar N.S.A.A., Boughton C.	2008	Proceedings of the 9th IASTED International Conference on Software Engineering and Applications, SEA 2008	CE1	--
Using A Line Of Code Metric To Understand Software Rework	Morozoff, E.	2010	Software, IEEE	CE1	--
Using Analogy To Predict Functional Regions On Genes	Pastor, J.A.; Koile, K.; Overton, G.C.	1991	System Sciences, 1991. Proceedings of the Twenty-Fourth Annual Hawaii International Conference on	CE1	--
Using Decision Trees To Predict The Certification Result Of A Build	Hassan, A.E.; Zhang, K.	2006	Automated Software Engineering, 2006. ASE '06. 21st IEEE/ACM International Conference on	CE1	--
Using Early Stage Project Data To Predict Change-Proneness	Ingram, C.; Riddle, S.	2012	Emerging Trends in Software Metrics (WETSoM), 2012 3rd International Workshop on	CE1	--
Using Hessian Locally Linear Embedding For Autonomic Failure Prediction	Xu Lu; Huiqiang Wang; Renjie Zhou; Baoyu Ge	2009	Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on	CE1	--
Using Model Transformation To Support Model-Based Test Coverage Measurement	Naslavsky L., Ziv H., Richardson D.J.	2008	Proceedings - International Conference on Software Engineering	CE1	--
Using Search-Based Metric Selection And Oversampling To Predict Fault Prone Modules	Vivanco, R.; Kamei, Y.; Monden, A.; Matsumoto, K.; Jin, D.	2010	Electrical and Computer Engineering (CCECE), 2010 23rd Canadian Conference on	CE1	--
Using Software Component Models And Services In Embedded Real-Time Systems	Frank Luders; Shoaib Ahmad; Faisal Khizer; Gurjodh Singh-Dhillon	2007	System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on	CE1	--
Using Sql Hotspots In A Prioritization Heuristic For Detecting All Types Of Web Application Vulnerabilities	Smith, B.; Williams, L.	2011	Software Testing, Verification and Validation (ICST), 2011 IEEE Fourth International Conference on	CE1	--
Using The Gini Coefficient For Bug Prediction In Eclipse	Giger E., Pinzger M., Gall H.	2011	IWPSE-EVOL'11 - Proceedings of the 12th International Workshop on Principles on Software Evolution	CE1	--
Using Variable-Mhz Microprocessors To Efficiently Handle Uncertainty In Real-Time Systems	Rotenberg, E.	2001	Microarchitecture, 2001. MICRO-34. Proceedings. 34th ACM/IEEE International Symposium on	CE1	--
Value Creation And Capture: A Model Of The Software Development Process	Little, T.	2004	Software, IEEE	CE1	--
Vector Space Analysis Of Software Clones	Grant, S.; Cordy, J.R.	2009	Program Comprehension, 2009. ICPC '09. IEEE 17th International Conference on	CE1	--
Ventricular Fibrillation Detection Using A Leakage/Complexity Measure Method	Moraes, J.C.T.B.; Blechner, M.; Vilani, F.N.; Costa, E.V.	2002	Computers in Cardiology, 2002	CE1	--
Versatile Prediction And Fast Estimation Of Architectural Vulnerability Factor From Processor Performance Metrics	Lide Duan; Bin Li; Lu Peng	2009	High Performance Computer Architecture, 2009. HPCA 2009. IEEE 15th International Symposium on	CE1	--
Video Quality Assessment Using Temporal Quality Variations And Machine Learning	Narwaria, Manish; Weisi Lin	2011	Multimedia and Expo (ICME), 2011 IEEE International Conference on	CE1	--
Wirelength Prediction For Fpgas	Pandit A., Akoglu A.	2007	Proceedings - 2007 International Conference on Field Programmable Logic and Applications, FPL	CE1	--
Ieee Standard Glossary Of Software Engineering Terminology		1990	IEEE Std 610.12-1990	CE1, CE2	--
2008 Acm/Ieee 30th International Conference On Software Engineering,Icse 2008	[No author name available]	2008	Proceedings - International Conference on Software Engineering	CE2	--
2012 1st International Workshop On Realizing Ai Synergies In	[No author name available]	2012	2012 1st International Workshop on Realizing AI Synergies in Software	CE2	--

Software Engineering, Raise 2012 - Proceedings			Engineering, RAISE 2012 - Proceedings		
30th International Conference On Software Engineering, Icse 2008 Co-Located Workshops - Proceedings Of The 4th International Workshop On Predictor Models In Software Engineering, Promise'08	[No author name available]	2008	Proceedings - International Conference on Software Engineering	CE2	--
30th International Conference On Software Engineering, Icse 2008 Co-Located Workshops - Proceedings Of The 6th International Workshop On Software Quality, Wosq'08	[No author name available]	2008	Proceedings - International Conference on Software Engineering	CE2	--
Comments, With Reply, On 'Predicting Source-Code Complexity At The Design State' By S. Henry And C. Selig	Coulter, N.S.	1990	Software, IEEE	CE2	--
Icsoft 2011 - Proceedings Of The 6th International Conference On Software And Database Technologies	[No author name available]	2011	ICSOFT 2011 - Proceedings of the 6th International Conference on Software and Database Technologies	CE2	--
Msr'11: Proceedings Of The 8th Working Conference On Mining Software Repositories, Co-Located With Icse 2011	[No author name available]	2011	Proceedings - International Conference on Software Engineering	CE2	--
Notice Of Violation Of Ieee Publication Principles Evaluation Of Gp Model For Software Reliability	Paramasivam, S.; Kumaran, M.	2009	2009 International Conference on Signal Processing Systems	CE2	--
Proceedings - 14th European Conference On Software Maintenance And Reengineering, Csmr 2010	[No author name available]	2011	Proceedings of the European Conference on Software Maintenance and Reengineering, CSMR	CE2	--
Proceedings - 1st International Symposium On Search Based Software Engineering, Ssbse 2009	[No author name available]	2009	Proceedings - 1st International Symposium on Search Based Software Engineering, SSBSE 2009	CE2	--
Proceedings And Companion To The Wikisym'07 Ismm'07, Hpc-Geco/Compframe'07 - 22nd Acm Sigplan Conference On Object-Oriented Programming Systems And Applications Companion , Oopsla'07	[No author name available]	2007	Proceedings of the Conference on Object-Oriented Programming Systems, Languages, and Applications, OOPSLA	CE2	--

APÊNDICE C – PARAMETRIZAÇÕES E DESVIOS PADRÕES DOS ERROS RELATIVOS MÉDIOS APRESENTADOS NO CAPÍTULO 5

Neste apêndice, são apresentados os Desvios Padrões dos Erros Relativos Médios apresentados no Capítulo 5 e os parâmetros utilizados em cada técnica que geraram tais valores. Para facilitar a correspondência com os valores das tabelas do capítulo, os valores de Desvio Padrão estão na mesma ordem em que os valores de Erro Relativo Médio aparecem.

Por questões de espaço, as informações foram abreviadas. Assim, P_i representa as parametrizações relativas a cada horizonte H_i, W representa a janela utilizada pela técnica Médias Móveis, SC é a condição de parada utilizada pelas técnicas do tipo Dinâmico, K é o valor de k das técnicas da família KNN, Euc representa a distância Euclidiana, Mht representa a distância de Manhatann, Avg representa a Média e Med representa a mediana.

Técnica	P1	H1	P2	H2	P3	H3	P4	H4	P5	H5	P6	H6	P7	H7	P8	H8	P9	H9	P10	H10
Médias Móveis	W=9	9,99	W=9	9,99	W=9	9,99	W=9	9,99	W=9	9,99	W=9	9,99	W=9	9,99	W=9	9,99	W=9	9,99	W=9	9,99
Local KNN	SC= 0,04	9,99	SC= 0,04	9,99	SC= 0,04	9,99	SC= 0,04	9,99	SC= 0,04	9,99	SC= 0,04	9,99	SC= 0,04	9,99	SC= 0,04	9,99	SC= 0,04	9,99	SC= 0,04	9,99
Global KNN	SC= 0,04	9,99	SC= 0,04	9,99	SC= 0,04	9,99	SC= 0,04	9,99	SC= 0,04	9,99	SC= 0,04	9,99	SC= 0,04	9,99	SC= 0,04	9,99	SC= 0,04	9,99	SC= 0,04	9,99
Reg. Lin.	N/A	9,99	N/A	9,99	N/A	9,99	N/A	9,99	N/A	9,99	N/A	9,99	N/A	9,99	N/A	9,99	N/A	9,99	N/A	9,99
Reg. Quad.	N/A	9,99	N/A	9,99	N/A	9,99	N/A	9,99	N/A	9,99	N/A	9,99	N/A	9,99	N/A	9,99	N/A	9,99	N/A	9,99
Local	SC= 0,04	9,99	SC= 0,04	9,99	SC= 0,04	9,99	SC= 0,04	9,99	SC= 0,04	9,99	SC= 0,04	9,99	SC= 0,04	9,99	SC= 0,04	9,99	SC= 0,04	9,99	SC= 0,04	9,99
Global	SC= 0,04	9,99	SC= 0,04	9,99	SC= 0,04	9,99	SC= 0,04	9,99	SC= 0,04	9,99	SC= 0,04	9,99	SC= 0,04	9,99	SC= 0,04	9,99	SC= 0,04	9,99	SC= 0,04	9,99
KNN	K=99 Euc Avg.	9,99	K=99 Mht Avg.	9,99	K=99 Euc Med.	9,99	K=99 Mth Med.	9,99	K=99 Euc Avg.	9,99	K=99 Mht Avg.	9,99	K=99 Euc Med.	9,99	K=99 Mth Med.	9,99	K=99 Euc Avg.	9,99	K=99 Euc Avg.	9,99
KNN-IR	K=99 Euc Avg.	9,99	K=99 Mht Avg.	9,99	K=99 Euc Med.	9,99	K=99 Mth Med.	9,99	K=99 Euc Avg.	9,99	K=99 Mht Avg.	9,99	K=99 Euc Med.	9,99	K=99 Mth Med.	9,99	K=99 Euc Avg.	9,99	K=99 Euc Avg.	9,99
KNN-IA	K=99 Euc Avg.	9,99	K=99 Mht Avg.	9,99	K=99 Euc Med.	9,99	K=99 Mth Med.	9,99	K=99 Euc Avg.	9,99	K=99 Mht Avg.	9,99	K=99 Euc Med.	9,99	K=99 Mth Med.	9,99	K=99 Euc Avg.	9,99	K=99 Euc Avg.	9,99