INFRAESTRUTURA COMPUTACIONAL PARA OBSERVAÇÃO DE EVOLUÇÃO DE SOFTWARE

Vitor Faria Monteiro

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientadores: Guilherme Horta Travassos
Marco Antônio Pereira Araújo

Rio de Janeiro
Agosto de 2011

INFRAESTRUTURA COMPUTACIONAL PARA OBSERVAÇÃO DE EVOLUÇÃO DE SOFTWARE

Vitor Faria Monteiro

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

_____

Prof. Guilherme Horta Travassos, D.Sc.

_____

Prof. Marco Antônio Pereira Araújo, D.Sc.

_____

Prof. Toacy Cavalcante de Oliveira, D.Sc.

_____

Prof. Márcio de Oliveira Barros, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
AGOSTO DE 2011

Monteiro, Vitor Faria

Infraestrutura Computacional para Observação de Evolução de Software / Vitor Faria Monteiro – Rio de Janeiro: UFRJ/COPPE, 2011.

X, 168 p.: il.; 29,7 cm.

Orientadores: Guilherme Horta Travassos

Marco Antônio Pereira Araújo.

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2011.

Referências Bibliográficas: p. 106-112.

1. Evolução de Software. 2. Simulação. 3. Engenharia de Software Experimental. 4. Sistemas de Software Orientados a Objetos. I. Travassos, Guilherme Horta, *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M. Sc.).

INFRAESTRUTURA COMPUTACIONAL PARA OBSERVAÇÃO DE EVOLUÇÃO DE SOFTWARE

Vitor Faria Monteiro

Agosto/2011

Orientadores: Guilherme Horta Travassos
Marco Antônio Pereira Araújo

Programa: Engenharia de Sistemas e Computação

A evolução de software é um fenômeno que está intimamente relacionado aos processos de desenvolvimento e manutenção e pode ser caracterizado pelas diversas modificações realizadas nos artefatos produzidos e mantidos ao longo do ciclo de vida do software. Uma maior compreensão sobre como os sistemas evoluem ao longo do tempo possibilita um maior controle sobre o processo de evolução. Desta forma, atividades de planejamento e gerenciamento podem ser melhor aplicadas sobre esse processo visando garantir maior qualidade às constantes modificações realizadas durante o ciclo de vida do software.

Esta dissertação apresenta um estudo sobre um modelo de observação de evolução de software proposto anteriormente, visando analisar e aumentar a validade deste modelo através de estudos experimentais. Para isso, inicialmente foram executados estudos secundários, baseados em revisão sistemática, com o objetivo de encontrar evidências adicionais na literatura técnica sobre os relacionamentos apresentados no modelo original. Os resultados das revisões permitiram aumentar a validade de construção do modelo, já que foram encontradas evidências adicionais reafirmando grande parte dos relacionamentos. Além disso, também foram encontradas evidências para um novo relacionamento, o que gerou a criação de um novo modelo de observação de evolução de software. Após a execução das revisões, foram executados estudos experimentais de simulação, utilizando dados de um projeto real, com o objetivo de avaliar os resultados obtidos com o novo modelo, comparando os resultados observados na realidade do projeto.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.).

COMPUTING INFRASTRUCTURE FOR OBSERVING SOFTWARE EVOLUTION

Vitor Faria Monteiro

August/2011

Advisors: Guilherme Horta Travassos
Marco Antônio Pereira Araújo

Department: Computer Science and Systems Engineering

Software evolution is a phenomenon intrinsically related to the software development and maintenance processes. Its main characteristic is the large amount of changes performed in the artifacts throughout the software life cycle. A greater understanding about how software evolves over time can enable a greater control over its evolution process. Thus, planning and management can be better applied to ensure more quality to the changes performed throughout the software life cycle.

This master dissertation presents a study about a previously proposed observation model regarding software evolution with the aim of analyze and improve its validity through experimental studies. For this, secondary studies based on systematic reviews were initially executed aiming to find additional evidence in the technical literature about the relationships described by the model. The systematic review's results allowed improving its construct validity due to the additional evidence found in the technical literature concerned with most of the original relationships present in the model. Moreover, additional evidence supporting a new relationship were found, which generate a new version of the software evolution observation model. After this, simulation studies using data from an industrial project were executed allowing assessing the results obtained with this new model's version by comparing them with the actual project status and with previous simulation results obtained with the original model.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

1 Introdução

1.1 Motivação

O termo evolução reflete um processo de mudança progressiva em atributos ou elementos que constituem a entidade em evolução (LEHMAN e RAMIL 2002). A experiência e a prática no desenvolvimento de software permitem a compreensão de que a mudança é uma característica intrínseca e praticamente inevitável dentro do processo de desenvolvimento de software.

A evolução de software tem sido reconhecida como uma das áreas mais problemáticas e desafiadoras no campo da Engenharia de Software, onde mais de 80% dos custos do ciclo de vida do desenvolvimento são gastos após a entrega do sistema. Entretanto, apesar de sua importância econômica, tem recebido relativamente pouca atenção no campo da Engenharia de Software. Além disso, geralmente, profissionais da academia e da indústria não tem informação e compreensão suficientes sobre o porquê e como os sistemas progressivamente mudam ao longo do tempo (MADHAVJI *et al*. 2006). Portanto, o processo de evolução de software deve ser melhor compreendido para ser tratado com planejamento e gerenciamento.

De acordo com LEHMAN e RAMIL (2002), quanto maior compreensão sobre o processo de evolução dos sistemas maior será a possibilidade de melhorar as metodologias, os processos e tecnologias de desenvolvimento e gerenciamento de software.

Nesse contexto, pode-se considerar que a área de evolução de software apresenta reais desafios para a pesquisa em Engenharia de Software, onde a elaboração de modelos e infraestruturas para observação da evolução do software são pontos de partida para uma melhor compreensão do processo de evolução.

Diante da importância do processo de evolução de software, este trabalho apresenta a continuidade de um estudo sobre um modelo de observação de evolução de software (ARAÚJO 2009), onde foi elaborado um modelo de simulação baseado em Dinâmica de Sistemas que possibilita simular o comportamento de sistemas ao longo de sucessivos ciclos de manutenção, proporcionando a oportunidade de se obter melhor compreensão de seu processo de evolução.

Uma das principais motivações dessa pesquisa foi dar continuidade ao estudo de ARAÚJO (2009) sobre o processo de evolução de software, tendo em vista os bons resultados apresentados por ele e a importância do processo para a área de Engenharia de Software.

1.2 Objetivos

Este trabalho tem como objetivo principal avaliar o modelo conceitual de observação de evolução de software, proposto por ARAÚJO (2009), através da verificação da validade conceitual dos relacionamentos apresentados no modelo e também realizar a atualização do modelo de Dinâmica de Sistemas (FORRESTER 1961) construído com base nestes relacionamentos, reavaliando sua viabilidade através de novos estudos de simulação.

Para alcançar o objetivo principal e também contribuir de outras formas para a área de evolução de software, foram definidos os seguintes objetivos secundários:

- Realizar um estudo baseado em revisão sistemática (KITCHENHAM *et al.* 2004; BIOLCHINI *et al.* 2005) através da reexecução dos protocolos de revisão definidos anteriormente (ARAÚJO 2009). O objetivo é encontrar evidências adicionais que reafirmem ou que sugiram modificações nos relacionamentos do modelo conceitual já construído, e desta forma, atualizar o corpo de conhecimento utilizado como base para construção do modelo;
- Realizar a atualização conceitual do modelo de observação de evolução de software com base nos resultados da revisão e também realizar alterações no modelo de Dinâmica de Sistemas com base na atualização do modelo conceitual;
- Realizar um estudo experimental de simulação com o novo modelo e comparar com os resultados da simulação obtidos com o modelo original. Para possibilitar a comparação, o mesmo conjunto de dados utilizado por ARAÚJO (2009) será utilizado;
- Realizar uma nova coleta de dados de um sistema de software orientado à objetos para executar um segundo estudo experimental de simulação e verificar se os resultados obtidos a partir da simulação estão coerentes com a situação real do sistema;
- Desenvolver uma nova infraestrutura computacional, baseada na plataforma Web, para dar maior suporte à execução do procedimento de simulação através da automatização do procedimento, e também possibilitar diferentes formas de visualização dos resultados da simulação, utilizando o modelo atualizado;
- Elaborar um conjunto de diretrizes que auxilie a utilização da infraestrutura apresentada, com recomendações sobre a realização da coleta de dados e sobre a execução do procedimento de simulação.

1.3 Metodologia de Trabalho

Para atingir os objetivos desta pesquisa foi definido um fluxo de trabalho, apresentado na Figura 1-1 descrevendo as atividades realizadas, formando assim a estrutura da metodologia aplicada.

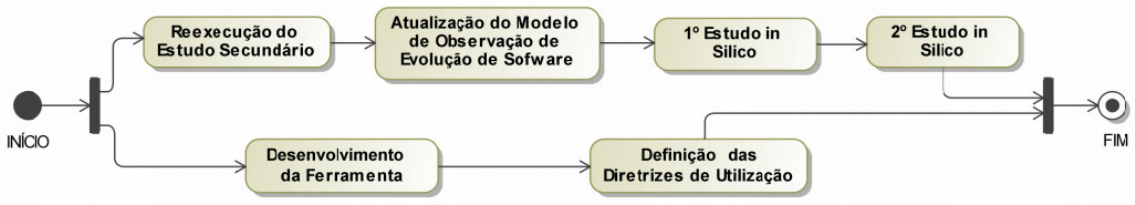


Figura 1-1 – Fluxo de trabalho representando a metodologia aplicada.

A reexecução do estudo secundário consistiu na utilização do protocolo de revisão sistemática definido por ARAÚJO (2009), para atualizar e dar maior abrangência aos resultados do estudo e assim dar maior validade conceitual ao modelo.

A atualização do modelo de observação de evolução de software consistiu na análise dos resultados obtidos pela revisão e posteriormente, na inclusão desses resultados como evidências adicionais para os relacionamentos apresentados no modelo original. A inclusão de evidências adicionais ao modelo original deu origem a um novo modelo conceitual de observação de evolução de software e também a um novo modelo de Dinâmica de Sistemas, que representa o modelo conceitual.

O 1º estudo *in silico* consistiu na execução de um estudo experimental de simulação utilizando o novo modelo de observação de evolução de software definido. Os dados utilizados nesse estudo foram os mesmos dados utilizados por ARAÚJO (2009) para possibilitar uma comparação entre os resultados obtidos a partir da simulação do modelo original e do novo modelo.

O 2º estudo *in silico* consistiu na execução de um novo estudo experimental de simulação utilizando o novo modelo de observação de evolução de software, mas dessa vez novos dados foram coletados do mesmo sistema utilizado no 1º estudo.

Em paralelo às atividades de pesquisa, uma ferramenta Web foi desenvolvida para automatizar a execução do procedimento de simulação do novo modelo, dando maior suporte à execução da simulação e interpretação dos resultados.

Após o desenvolvimento da ferramenta, foi definido um conjunto de diretrizes para apoiar na utilização do modelo e na execução do procedimento de simulação.

1.4 Organização

Além do capítulo de introdução, este trabalho possui mais cinco capítulos organizados da seguinte maneira:

- **Capítulo 2 – Conceitos relacionados à Evolução de Software:** apresenta os conceitos utilizados no contexto deste trabalho como, evolução de software, decaimento de software, as Leis de Evolução de Software, simulação (algumas técnicas e ferramentas) e o modelo de observação de evolução de software estudado;
- **Capitulo 3 – Atualização do Modelo de Observação de Evolução de Software:** apresenta a reexecução do estudo secundário baseado em revisão sistemática e a atualização do modelo realizada através dos resultados da revisão;
- **Capítulo 4 – Estudos Experimentais de Simulação em Evolução de Software:** apresenta as modificações realizadas no modelo de Dinâmica de Sistemas devido à atualização do modelo, resultados das simulações executadas com o novo modelo e também uma comparação entre os resultados obtidos através da simulação do modelo original e do novo modelo;
- **Capítulo 5 – Diretrizes para Utilização do Modelo de Observação de Evolução de Software:** apresenta as diretrizes definidas para a utilização do modelo e diretrizes para guiar a execução do procedimento de simulação através da apresentação da ferramenta Web desenvolvida;
- **Capítulo 6 – Conclusões Finais:** apresenta as conclusões, contribuições do trabalho, limitações e discute algumas perspectivas de trabalhos futuros.

1.5 Síntese dos Resultados Alcançados

Dentre os resultados alcançados neste trabalho, destacam-se:

- Atualização do modelo de observação de evolução de software através das evidências adicionais encontradas sobre os relacionamentos presentes no modelo original;
- Adição de um novo relacionamento ao modelo original, resultando em um novo modelo de observação de evolução de software;
- Atualização e melhorias no modelo de Dinâmica de Sistemas correspondente ao modelo conceitual, possibilitando a observação de novos comportamentos;

- Automatização do procedimento de simulação através de uma ferramenta Web.

2 Conceitos Relacionados à Evolução de Software

Neste capítulo, serão apresentados os conceitos sobre evolução de software, decaimento de software, as Leis de Evolução de Software (LEHMAN 1980), conceitos sobre técnicas de simulação e a apresentação do modelo de observação de evolução de software analisado neste trabalho.

A evolução de software é um fenômeno que ocorre constantemente em softwares que estão relacionados a algum processo dinâmico do mundo real e, geralmente esse fenômeno é pouco compreendido, levando algumas vezes ao decaimento da qualidade de software. As Leis de Evolução de Software representam uma importante contribuição para a comunidade de pesquisa em evolução de software no sentido de se alcançar uma maior compreensão sobre o fenômeno (SCACCHI 2003). De acordo com (LEHMAN 1997), a simulação é uma ferramenta apropriada para auxiliar na obtenção de uma maior compreensão sobre esse fenômeno.

2.1 Evolução de Software

Em geral, um sistema de software é desenvolvido para solucionar problemas ou automatizar processos do mundo real que está constantemente sofrendo mudanças por diversos motivos, principalmente devido a interesses econômicos. Para acompanhar o dinamismo do mundo real, o processo de desenvolvimento de software deve refletir essas mudanças adequadamente, tanto no sentido de atender às novas necessidades dos usuários, quanto no sentido de garantir a qualidade da estrutura do software. De acordo com LEHMAN (1980), a qualidade de um sistema entrará em declínio caso não sejam tomadas medidas para sua manutenção diante das mudanças do ambiente operacional.

Na literatura técnica existem diferentes interpretações sobre o termo evolução de software, e muitas vezes, existe uma confusão entre os termos manutenção e evolução de software.

Segundo a IEEE (1990), a definição de manutenção está dividida em três categorias: Adaptativa, Corretiva e a Perfectiva. A manutenção adaptativa se baseia na adição de novas funcionalidades ao sistema ou na adaptação do sistema devido alguma alteração em seu ambiente para manter a utilidade do software. A manutenção corretiva tem o objetivo de corrigir defeitos no software. E por fim, a manutenção perfectiva, tem como objetivo melhorar a manutenibilidade e o desempenho do sistema aplicando conceitos de re-engenharia.

De acordo com ISO/IEC 12207 (2008), manutenção é um dos principais processos do ciclo de vida do desenvolvimento, onde o produto passa por modificações, no código e na documentação associada, devido a problemas ou necessidades de melhoria. O objetivo é modificar o produto de software já existente preservando sua integridade.

Alguns autores (KEMERER e SLAUGHTER apud. ARAÚJO 2009) fazem uma distinção entre manutenção e evolução de software. A manutenção é entendida como a correção de erros e a implementação de modificações necessárias para permitir a um sistema existente executar novas tarefas e executar antigas sob novas condições, enquanto a evolução trata o comportamento dinâmico dos sistemas, como eles são mantidos e expandidos ao longo de seu ciclo de vida. Ou seja, a manutenção se refere às atividades que acontecem a qualquer época após a implementação de um novo projeto e a evolução é definida pelo exame do comportamento dinâmico das características do sistema, isto é, pelo estudo de como elas mudam ao longo do tempo.

No contexto de pesquisa na área Engenharia de Software o termo evolução pode ser interpretado de duas maneiras distintas (LEHMAN *et al.* 2000). A visão mais difundida considera que os tópicos importantes sobre evolução são aqueles que se preocupam com meios pelos quais a evolução será gerenciada e implementada, onde os meios significam os mecanismos e ferramentas pelos quais a evolução poderá ser obtida de acordo com um planejamento. Nesse caso, o foco é compreender como ocorre o processo da evolução de software (MADHAVJI *et al.* 2006).

Outra visão menos difundida, mas igualmente importante, busca uma compreensão da natureza do fenômeno da evolução, suas causas, propriedades e características, suas consequências, impactos, gerenciamento e controle (LEHMAN 2003). Nesse caso, o foco é investigar o que é a evolução de software e o porquê ela acontece (MADHAVJI *et al.* 2006).

Portanto, pode-se observar que o conceito de manutenção de software é um termo amplo que envolve um processo específico dentro do ciclo de vida do desenvolvimento de software, enquanto o termo evolução de software representa um fenômeno que ocorre como consequência do processo de manutenção.

De acordo com PRESSMAN (2009), a mudança conduz o processo de evolução e ocorre quando defeitos são encontrados, quando o software é adaptado para um novo ambiente, quando o cliente solicita novas características ou funcionalidades ou quando a aplicação passa por um processo de re-engenharia.

Desta forma, neste trabalho a evolução de software será considerada como um fenômeno que ocorre devido ao processo de manutenção de software.

Para observar a evolução de um sistema de software o ideal seria analisar dados correspondentes a atividade de manutenção evolutiva, entretanto, foi identificado por ARAÚJO (2009) que a classificação dos dados de acordo com as categorias de manutenção seria muito complexa e que geralmente os dados extraídos de repositórios de sistemas reais não consideram essa distinção. Portanto, assim como em ARAÚJO (2009), para observar a evolução de software, o processo de manutenção será tratado como um todo já que, a princípio, todas as modificações influenciam na evolução de sistemas de software.

Existem diversos motivos que ocasionam mudanças em um software. Além das modificações tecnológicas; revelam-se novas necessidades dos *stakeholders*; mudanças na legislação, regras e regulamentações, que ocorrem no domínio em que o software está inserido. Outra fonte importante de mudanças em um software é o aprendizado do usuário após a implantação e o uso do sistema. Com a utilização do sistema no cotidiano da organização, o usuário passa a compreender melhor seu funcionamento e também a perceber novas possibilidades sobre como o sistema poderia otimizar determinada atividade do seu fluxo de trabalho. Em alguns casos esse aprendizado causa modificações não só no sistema, mas também no domínio dentro do qual ele se aplica. Desde os primeiros estudos executados sobre a evolução de software (LEHMAN 1980) foi demonstrado que esse é um fenômeno que pode ser observado, medido e analisado com o sistema de realimentação (*feedback*) desempenhando o papel mais importante na determinação do comportamento.

Diversos autores ressaltam a importância do processo de manutenção dentro do ciclo de vida do desenvolvimento de software, entretanto, esta é uma atividade que geralmente não é tratada de maneira planejada e gerenciada. De acordo com ARAÚJO (2009), geralmente a manutenção em sistemas de software acontece de maneira relativamente desorganizada, o que conduz naturalmente a deterioração da estrutura de sistemas de software.

Diante da contínua evolução que os sistemas que operam no mundo real sofrem, seja ela para correção, adaptação, melhoria ou extensão, é claramente necessário garantir que as alterações reflitam adequadamente as necessidades do domínio em que o sistema está inserido, e para isso, é também evidente que esta evolução deve ser planejada, direcionada e gerenciada (LEHMAN 2002).

2.2 Decaimento de Software

De acordo com EICK (1999 *apud*. ARAÚJO 2009), o decaimento de software é caracterizado pela deterioração da estrutura de um sistema em evolução, afetando sua qualidade e dificultando a realização de novas manutenções.

O software, por ser um elemento de um sistema lógico e não de um sistema físico, possui algumas características que o tornam diferente de outros produtos que o ser humano constrói. O software não se desgasta, mas se deteriora (PRESSMAN 2005). Esta é uma característica, intrínseca à natureza do software, que está intimamente relacionada ao processo de evolução de software.

De acordo com PRESSMAN (2009), a Figura 2-1 exibe uma curva ideal mostrando como seria, em teoria, a curva de taxa de falhas de um software, onde os defeitos não capturados causariam altas taxas de falhas no início de vida do sistema e assim estes seriam corrigidos (sem introduzir novos defeitos) e a curva de taxa de falhas diminuiria.

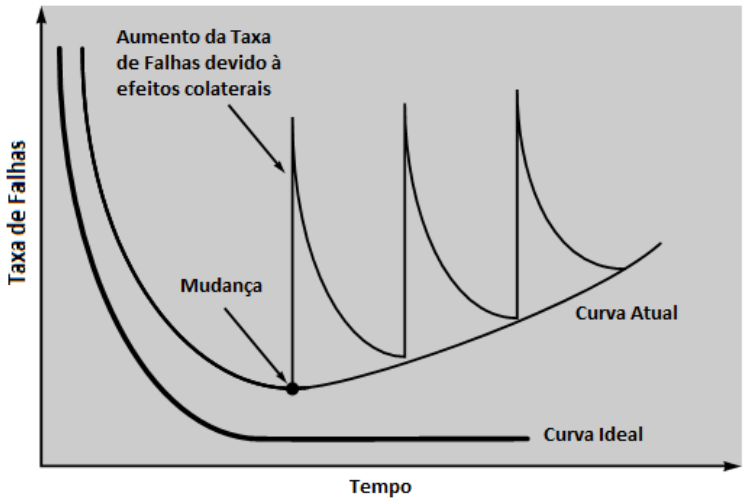


Figura 2-1 – Curvas de taxa de falhas de software (adaptada de PRESSMAN 2009).

Além da curva ideal, a Figura 2-1 mostra também a curva atual, que exibe um comportamento mais próximo da realidade. Durante sua vida, o software sofrerá modificações e, à medida que elas aconteçam, é possível que defeitos sejam introduzidos, causando o aumento da taxa de falhas, como mostrado na Figura 2-1. Antes que a curva retorne para seu estado estável, outra modificação é solicitada, causando o aumento na taxa de falhas novamente. Lentamente, o nível mínimo da taxa de falhas começa a aumentar – o software está deteriorando devido às modificações.

Os sistemas de software estão em constante evolução (LEHMAN 2000), ou seja, constantemente sofrendo modificações, portanto caso não sejam tomadas

medidas que evitem a deterioração do sistema, a evolução de software conduzirá naturalmente o decaimento do software.

2.3 Leis de Evolução de Software

O trabalho de LEHMAN (1980) é um dos precursores da área de evolução de software e descreve um sistema em termos dos caminhos em que se relaciona com o ambiente em que opera, sendo sucedido por vários trabalhos que relatam a evolução de sistemas, iniciando na década de 70 (ARAÚJO 2009). Nos trabalhos foram definidas oito leis, conhecidas como Leis de Evolução de Software, que de acordo com LEHMAN *et al.* (1998) focam primeiramente no entendimento de como os sistemas de software mudam ao longo do tempo. A Tabela 2-1 apresenta as Leis de Evolução de Software definidas por LEHMAN (1980).

Tabela 2-1 – Leis de Evolução de Software.

Enunciado	Lei de Evolução de Software
I – Mudança Contínua (Postulada em 1974)	Um sistema que está em uso sofre mudança constante ou se torna progressivamente menos útil. A mudança ou o processo de decaimento continua até que seja mais barato substituir o sistema com uma versão recriada.
II – Incremento da Complexidade (Postulada em 1974)	Um programa em desenvolvimento é continuamente modificado e esta mudança constante introduz cada vez mais complexidade no produto, deteriorando-lhe a estrutura; se não for desenvolvida nenhuma atividade explícita do controle da complexidade, a manutenção do produto pode deixar de ser possível e ele se tornar inútil.
III – Autorregulação (Postulada em 1974)	Lei Fundamental da Evolução do Produto; o processo de evolução do produto é uma dinâmica autorregulada com tendências estatísticas determináveis e invariâncias; sistemas de software exibem comportamentos regulares e tendências que podem ser medidas e previstas.
IV – Conservação da Estabilidade Organizacional (Postulada em 1978)	Durante o ciclo de vida de um produto a taxa de atividade global em um produto em evolução é estatisticamente invariante; atributos organizacionais, como produtividade, não exibem grandes flutuações; recursos e resultados alcançam um nível ótimo e adicionar mais recursos não muda significativamente os resultados.
V – Conservação da Familiaridade (Postulada	Durante o ciclo de vida de um produto, o conteúdo de cada versão é estatisticamente invariante; o conteúdo de sucessivas

em 1978)	versões de programas em evolução (mudanças, adições, exclusões) é estatisticamente invariante; após um tempo, o efeito de versões de manutenções sucessivas faz pouca diferença na funcionalidade geral.
VI – Crescimento Contínuo (Postulada em 1978)	O conteúdo funcional de um sistema deve ser continuamente incrementado para manter a satisfação do usuário ao longo do ciclo de vida da aplicação.
VII - Declínio da Qualidade (Postulada em 1994)	A qualidade de um sistema entrará em declínio a menos que seja rigorosamente mantida e adaptada às mudanças do ambiente operacional.
VIII – Sistema de Realimentação (Postulada em 1972 e modificada em 1996)	O processo de evolução de um sistema constitui em realimentação em multinível, multi-interação e multiagente do sistema e deve ser tratado de forma a alcançar significativas melhorias.

Apesar de terem sido postuladas há mais de 20 anos atrás, muitos estudos deram continuidade às pesquisas sobre as leis e comprovaram que elas são aplicáveis nos dias atuais (LEHMAN *et al.* 1998, Projeto FEAST/1, Projeto FEAST/2).

As Leis de Evolução de Software foram inicialmente definidas para serem aplicadas a sistemas classificados como *E-Type* (LEHMAN e RAMIL 2001). Esta classificação foi originalmente definida como sendo um sistema que automatiza uma atividade humana; outra definição seria de um sistema ativamente usado e embutido em um domínio do mundo real (LEHMAN e RAMIL 2003). Portanto, neste trabalho será apresentado um estudo sobre a evolução de um sistema de software classificado como *E-Type System.*

As Leis de Evolução de Software foram a base para a construção do modelo de observação de evolução de software proposto por ARAÚJO (2009), que é objeto de estudo deste trabalho. O modelo, baseado em evidências extraídas da literatura técnica de Engenharia de Software, é apoiado por técnicas de Dinâmica de Sistemas com o objetivo de possibilitar a simulação de como sistemas de software orientados a objetos se comportam ao longo de sucessivos ciclos de manutenção, de forma a proporcionar uma melhor compreensão do processo de evolução.

2.4 Simulação

A Simulação é um processo baseado na elaboração de um modelo representando um sistema real ou hipotético e na condução de experimentos com a finalidade de entender o comportamento de um sistema ou avaliar sua operação (SHANNON 1998). Simulação representa uma gama variada de métodos e aplicações que reproduzem o comportamento de sistemas reais, usualmente utilizando-se de ferramentas computacionais (KELLNER *et al.* 1999).

De acordo com (KELLNER 1998), o objetivo da simulação é examinar como o sistema se comporta durante um período de tempo e também analisar o comportamento de sistemas complexos, onde fatores como tempo, segurança, disponibilidade de recursos e pessoas são críticos. Para atingir esse objetivo, o modelo de simulação deve providenciar facilidades para representar o estado atual do sistema e várias pré-condições que, se satisfeitas, irão resultar num estado futuro.

O modelo de um sistema deve possuir uma representação para as características reais e também representar alguma dinâmica ou um fenômeno deste sistema (KELLNER 1999). Outro fator muito importante para o processo de simulação são os dados utilizados para guiar o modelo. A simulação de um processo será efetiva somente se o modelo e os dados utilizados para guiar o modelo refletirem precisamente o mundo real (CHRISTIE 1999). Portanto, a modelagem do sistema e a coleta dos dados para guiar o modelo são etapas fundamentais para que os resultados obtidos através do processo de simulação façam sentido diante de uma perspectiva real.

Uma das principais motivações para desenvolver um modelo de simulação é que esta abordagem representa uma maneira econômica para obter *insights* importantes quando o custo, o risco ou a logística para manipular o sistema real de interesse são elevados (KELLNER 1999). Modelos de simulação permitem visualizar e quantificar modelos mentais implícitos sobre as causas que governam os comportamentos de um processo sob observação e desta forma dar suporte para o entendimento, análise, predição e tomada de decisão (MULLER e PFAHL 2008).

Por vários anos, o processo de simulação tem sido aplicado em diversas áreas da ciência e nas últimas décadas vem ganhando grande importância na comunidade de Engenharia de Software, principalmente na área de simulação de processos de software. De acordo com (ZHANG *et al.* 2008), nas últimas duas décadas, simulação emergiu como uma ferramenta efetiva para auxiliar a avaliação e o gerenciamento de mudanças em projetos de software.

A simulação de processos foi introduzida no domínio de Engenharia de Software no final da década de 80 com o trabalho pioneiro de ABDEL-HAMID e MADNICK (1991). De acordo com (BARROS *et al.* 2000), neste trabalho foi desenvolvido um modelo de processo de software formalizando o efeito de políticas, procedimentos e ações gerenciais tomadas durante o processo de desenvolvimento. O modelo, desenvolvido com a técnica de Dinâmica de Sistemas (FORRESTER 1961), se divide em quatro seções, envolvendo o gerenciamento de recursos humanos, a produção de software, o planejamento e o controle do projeto.

No contexto da Engenharia de Software Experimental, o processo de simulação pode ser classificado em duas categorias de estudos: *In Virtuo* e *In Silico.* Os estudos *In Virtuo* são caracterizados quando os participantes interagem com um modelo computacional da realidade, ou seja, o comportamento do ambiente nos quais os indivíduos interagem é descrito através de um modelo computacional. Por outro lado, os estudos *In Silico* são caracterizados quando os participantes e o ambiente do estudo são descritos totalmente através de modelos computacionais, onde a interação humana é reduzida ao máximo (TRAVASSOS e BARROS 2003).

2.4.1 Técnicas de Simulação

Existem diversas técnicas de simulação que podem ser aplicadas em processos de Engenharia de Software para investigar comportamentos de um processo e a escolha de uma delas deve estar de acordo com o objetivo da simulação, ou seja, o que se deseja modelar, o propósito da simulação, as questões a serem analisadas, o escopo, os resultados desejados e outras características (KELLNER 1999).

De acordo com MULLER e PFAHL (2008), em geral, podem-se destacar quatro diferenças importantes entre as técnicas de simulação. A primeira é a diferença entre simulação Estocástica e Determinística. Modelos de simulação que contém componentes probabilísticos são chamados de estocásticos e aqueles que não possuem são chamados de determinísticos. No caso de um modelo determinístico, para um mesmo conjunto de valores de entrada, os resultados da simulação serão sempre os mesmos. Para um modelo estocástico, os resultados da simulação podem variar dependendo da variação estocástica (aleatória ou baseada em algum algoritmo genético, por exemplo) dos valores de entrada ou das variáveis intermediárias do modelo.

A segunda é a diferença entre simulação Estática e Dinâmica. Modelos de simulação estática capturam a variação dos parâmetros do modelo em um único ponto

no tempo, enquanto modelos de simulação dinâmicos capturam o comportamento dos parâmetros do modelo ao longo de um período específico do tempo.

A terceira é a diferença entre simulação Contínua e Discreta (orientada por eventos). Nesse caso, a diferença está na maneira como o estado interno do modelo é calculado. Modelos de simulação contínua atualizam os valores das variáveis que representam o estado do modelo em passos de tempo equidistantes com base em um conjunto bem definido de equações formalizando o modelo. A Dinâmica de Sistemas (FORRESTER 1961) é a técnica de simulação contínua mais popular e também pode ser considerada a mais utilizada no domínio da Engenharia de Software. De acordo com KNOP (2009), a Dinâmica de Sistemas facilita o entendimento e a comunicação da estrutura do processo a humanos. Os adeptos advogam que o entendimento do comportamento do sistema e da estrutura é o principal valor agregado ao uso da Dinâmica de Sistemas como ferramenta de predição. Modelos de simulação discreta atualizam as variáveis do modelo com base nos novos eventos que ocorrem. Existem diferentes tipos de técnicas de simulação orientada por eventos e a mais utilizada é a Simulação Discreta de Eventos (MULLER e PFAHL 2008).

A quarta é a diferença entre simulação Quantitativa e Qualitativa. A simulação quantitativa requer que os valores dos parâmetros dos modelos sejam especificados como números reais ou inteiros. O mais importante pré-requisito para este tipo de simulação é a disponibilidade de dados empíricos de qualidade e quantidade suficientes. Por outro lado, a simulação qualitativa é uma abordagem útil caso o objetivo seja entender o padrão de comportamento geral de um sistema dinâmico ou quando conclusões devem ser feitas a partir de uma quantidade pequena de dados.

De acordo com os resultados da revisão sistemática apresentados por ZHANG *et al.* (2008), os três métodos de modelagem de simulação mais utilizados em processos de software eram: Dinâmica de Sistemas, Simulação Discreta de Eventos e Simulação Baseada em Conhecimento, assim como mostrado na Figura 2-2.

Método	1998	1999	2000	2002	2003	2004	2005	2006	2007
SD	●	●	●	●	●	●	●	●	●
DES	●	●	●	●	●	●	●	●	●
SBS	●		●	●	●				●
KBS	●		●	●					
QSIM				●		●		●	●
RPG					●	●	●		
ABS							●	●	
DTS						●			
Total	4	2	4	5	4	5	4	4	4

SD: Dinâmica de Sistemas
SBS: Simulação baseada em estados
RPG: jogo de RPG
QSIM: Simulação Qualitativa
DES: Simulação Discreta de Eventos
KBS: Simulação baseada em conhecimento
ABS: Simulação baseada em agentes
DTS: Simulação de Tempo Discreto

Figura 2-2 – Métodos de modelagem de simulação (adaptado de ZHANG *et al.*, 2008).

Como se pode observar, alguns estudos aplicaram mais de um método de modelagem, ou seja, integraram mais de um método, o que caracteriza um modelo híbrido. É possível notar também que os métodos mais utilizados são: Dinâmica de Sistemas (SD) e Simulação Discreta de Eventos (DES).

2.4.1.1 Dinâmica de Sistemas

A Dinâmica de Sistemas foi introduzida por FORRESTER (1961) como um método para modelar e analisar o comportamento de sistemas complexos, formados por diversas variáveis que se relacionam de forma não linear. As diversas variáveis presentes no sistema exercem influência umas sobre as outras, formando laços de realimentação e determinando o comportamento do sistema.

Os modelos de Dinâmica de Sistemas são representados por um conjunto de equações diferenciais que são resolvidas através de integração numérica. As variáveis do modelo, que representam seu estado, são chamadas de *levels (stocks* ou variáveis de estado). Essas variáveis podem ser consideradas como *containers* ou reservatórios que acumulam alguma entidade tangível (por exemplo, pilhas de papel) ou intangível (por exemplo, número de defeitos), representada por algum atributo contável. As quantidades acumuladas em reservatório são reguladas através de válvulas contidas em canais de entrada e de saída. Nos modelos de Dinâmica de Sistemas, as variáveis do tipo *rate* (taxa ou fluxo) representam essas válvulas. As *rates* são representadas por equações. Essas variáveis podem depender de *levels*, constantes ou variáveis

auxiliares, que são utilizadas para desmembrar equações mais complexas (MULLER e PFAHL 2008).

Segundo Barros *et al.* (2000), matematicamente, a taxa representa a derivada parcial do nível do repositório em relação ao tempo, representando os aumentos e reduções do nível do repositório através de uma equação.

A integração numérica em Dinâmica de Sistemas implementa o cálculo integral, apresentado na Equação 1, para determinar o *level* no tempo *t*, com base em suas taxas de entrada e saída (MADACHY 2008):

$$Level\ (t + dt) = Level(t) + \int_{t}^{t+dt} [rate\_in(t) - rate\_out\ (t)]\ dt \quad \text{(Eq. 1)}$$

O parâmetro *dt* corresponde o tempo de incremento escolhido para a execução da simulação. A maioria das ferramentas poupa o modelador da construção das equações através de representação diagramática da modelagem e as equações são produzidas automaticamente (MADACHY 2008).

2.4.1.2 Simulação Discreta de Eventos

A simulação discreta de eventos é um método excelente para capturar tarefas de processos bem especificados, considerações sobre enfileiramento e cronograma de atividades, bem como ramificação de processos baseado em atributos de entidade. Entretanto, a crescente complexidade de processos de projetos de engenharia tem mostrado várias limitações deste paradigma (RAFFO et al., 1999).

As limitações críticas estão relacionadas com a dificuldade de capturar o impacto de fatores de mudanças, tais como produtividade, conhecimento e experiência da equipe ou pressão de cronogramas, entre outras características e, também, na dificuldade de fornecer um *feedback* em tempo real para o processo. Essas características devem permitir que tomadores de decisão analisem o impacto de questões relacionadas ao sistema que são particularmente importantes quando o processo é baseado no conhecimento de modelagem do desenvolvedor (RAFFO et al., 1999).

2.4.1.3 Modelos Híbridos

Um modelo híbrido é definido pela utilização de mais de um método na modelagem para simulação e o objetivo desses modelos fazerem a combinação de

mais de um método é de suprir as limitações de se aplicar um único método e capturar de forma mais realística processos de software reais e complexos.

De acordo com ZHANG *et al.* (2008), a utilização de métodos híbridos têm atraído interesse, entretanto ainda não é largamente utilizado, devido ao aumento da complexidade da modelagem para integrar de diferentes técnicas de modelagem.

De acordo com LEHMAN (1997), uma abordagem de simulação baseada em Dinâmica de Sistemas é mais apropriada para investigação do fenômeno de evolução de software do que uma abordagem analítica baseada em teoria de controle. O modelo de observação de evolução de software proposto por ARAÚJO (2009) utiliza a técnica de Dinâmica de Sistemas para a simulação. Como o foco deste trabalho não é investigar a técnica de simulação utilizada, a Dinâmica de Sistemas continuará sendo a ferramenta para a construção dos modelos apresentados neste trabalho.

2.4.2 Modelagem com Dinâmica de Sistemas

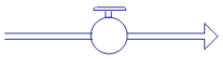
Nesta seção serão apresentados detalhes sobre a modelagem utilizando a técnica de Dinâmica de Sistemas através do apoio das ferramentas VENSIM® PLE (VENSIM 2011) e *Illium* (BARROS *et al.* 2000).

2.4.2.1 A Ferramenta Vensim® PLE

Vensim® PLE (*Personal Learning Edition*) é uma versão gratuita para uso pessoal e educacional da ferramenta de simulação Vensim®, fornecida pela empresa Ventana Systems, Inc. É uma ferramenta de modelagem visual que permite a concepção, documentação, simulação, análise e otimização de modelos de Dinâmica de Sistemas e também fornece uma maneira simples e flexível de construir modelos de simulação a partir de *loops* causais ou diagramas de estoque e fluxo. Além disso, oferece uma estação de trabalho que suporta representações do modelo tanto textuais quanto gráficas (VENSIM 2003a). De acordo com MULLER e PFAHL (2008), os símbolos utilizados pela ferramenta para a representação gráfica dos modelos seguem o padrão para modelagem em Dinâmica de Sistemas.

De acordo com (MADACHY 2008), as notações e os elementos utilizados pela ferramenta para modelagem de Dinâmica de Sistemas são apresentados na Tabela 2-2.

Tabela 2-2 – Elementos da modelagem de Dinâmica de Sistema utilizados em Vensim® (VENSIM 2003a).

Elemento	Notação	Descrição
Level (*Stock* ou variável de estado)		Um *level* é um acumulador ao longo do tempo.
Source/Sink		*Sources* e *sinks* indicam que fluxo vem a partir de ou vão para alguma fonte externa ao processo, respectivamente. Eles representam fontes ou repositórios infinitos que não são especificados no modelo.
Rate		*Rates* são chamadas também de fluxos. Elas representam a ação em um sistema. Causam as mudanças nas variáveis *level* e podem representar decisões ou regras.
Variáveis auxiliares ou Constantes	Nome da variável	São utilizadas para ajudar na elaboração da estrutura de *levels* e *rates,* contribuindo para desmembrar equações mais complexas.
Ligações de Informação		São utilizados para representar fluxo de informação. Podem representar *loops* de realimentação entre os elementos do modelo.

As variáveis do tipo *level* são representadas como caixas, enquanto as *rates* são representadas como válvulas com setas de duas linhas conectadas com as caixas. As constantes e variáveis auxiliares são representadas simplesmente por seus nomes e as setas de uma linha representam o fluxo de informação.

A definição de uma função é feita através de um editor de equação baseado em texto. O editor fornece uma janela de entrada para especificar a função exata a ser utilizada e também os campos para especificar as unidades das variáveis. Além disso, executa uma verificação automática da sintaxe e da consistência da equação (MULLER e PFAHL 2008).

A ferramenta Vensim® PLE foi utilizada neste trabalho com dois propósitos. O primeiro foi comparar os resultados da simulação obtidos através das ferramentas Vensim® PLE e *Illium* (BARROS *et al.* 2000) utilizando o novo modelo de observação de evolução de software que será apresentado posteriormente. O objetivo dessa comparação é verificar a coerência dos resultados apresentados pela ferramenta *Illium*

e assim, garantir maior confiabilidade aos resultados apresentados e dar maior validade ao estudo. O segundo propósito foi apresentar o modelo de Dinâmica de Sistemas através da notação gráfica fornecida pela ferramenta para facilitar a compreensão do modelo.

A escolha da ferramenta Vensim® PLE se deu pelo motivo de existir um grande suporte fornecido pelo distribuidor da ferramenta, através da disponibilização de documentação da ferramenta, guia para utilização e para modelagem. Além disso, o fato da ferramenta ser gratuita e amplamente utilizada tanto na academia quanto na indústria, influenciou em sua escolha.

2.4.2.2 A Ferramenta *Illium*

Illium é uma ferramenta desenvolvida por BARROS *et al.* (2000) que permite a construção, simulação e avaliação de modelos dinâmicos de projetos de desenvolvimento de software. Estes modelos são descritos utilizando-se a técnica de modelagem e simulação contínua, denominada Dinâmica de Sistemas.

A ferramenta define uma linguagem própria para representação dos modelos e também possui um interpretador para esta linguagem, sendo capaz de avaliar o comportamento de modelos dinâmicos de projetos de desenvolvimento de software. Além disso, a ferramenta possui diversos mecanismos de simulação e avaliação do comportamento dos modelos, como simulação no tempo e simulação estocástica (BARROS *et al.* 2000).

A construção de modelos dinâmicos nessa ferramenta é realizada através de uma descrição textual, utilizando a linguagem desenvolvida especificamente para esta tarefa. A sintaxe da linguagem é apresentada na Tabela 2-3, onde as palavras reservadas da linguagem estão destacadas em negrito.

Tabela 2-3 – Sintaxe da linguagem definida por *Illium* (BARROS *et al.* 2000).

SPEC DT <passo de simulação>
STOCK <*nome do repositório***> <***expressão***>;**
RATE *(Origem, Destino) <nome da taxa>* ***<expressão>;***
Origem = <nome do repositório> ou **SOURCE;**
*Destino= <nome do repositório>*ou **SINK;**
PROC <nome do processo**> <***expressão***>;**

De acordo com BARROS *et al.* (2000), o comando **SPEC DT** define a variação do tempo em cada passo de simulação. O comando **STOCK** permite a criação de um repositório, indicando seu nome e uma expressão, que determina o nível inicial do

repositório. O nome de um elemento (repositório, taxa ou processo) deve ser único dentre os elementos do modelo, sendo utilizado como referência para seu valor nas expressões. Na ferramenta Vensim® PLE, esse comando corresponde ao elemento *Level*, representado pelo símbolo de uma caixa.

O comando **RATE** permite a criação de uma taxa (fluxo), indicando seu repositório de origem, seu repositório destino, seu nome e sua expressão. Esta expressão indica a variação instantânea dos níveis dos repositórios associados à taxa. Em cada passo de simulação, o nível do repositório de origem será decrescido do valor resultante da expressão da taxa. De forma similar, o nível do repositório destino será acrescido deste montante (BARROS *et al*. 2000). Na ferramenta Vensim® PLE, esse comando corresponde ao símbolo representado pela seta de duas linhas com uma válvula, apresentado na terceira linha da Tabela 2-2.

O repositório origem de uma taxa pode ser um provedor universal, indicado pela palavra reservada **SOURCE**. Quando uma taxa está associada a um provedor universal, nenhum repositório do modelo terá seu nível debitado do valor da taxa. Da mesma forma, o repositório destino de uma taxa pode ser um receptor universal, indicado pela palavra reservada **SINK**, (BARROS *et al.* 2000). Na ferramenta Vensim® PLE, os comandos **SOURCE** e **SINK** correspondem ao símbolo da nuvem, apresentado na segunda linha da Tabela 2-2.

O comando **PROC** permite a definição de um processo, indicando seu nome e sua expressão. A expressão de um processo calcula um valor intermediário que pode ser utilizado em expressões de outros processos ou taxas do modelo (BARROS *et al.* 2000). Na ferramenta Vensim® PLE, esse comando corresponde às variáveis auxiliares ou constantes.

Além disso, de acordo com BARROS *et al.* (2000), as expressões utilizadas na especificação dos repositórios, taxas e processos, podem conter operadores algébricos (soma, subtração, produto, divisão e potência), operadores lógicos (e, ou, negação, teste de condição), operadores relacionais (maior, menor, maior ou igual, menor ou igual, igual e diferente) e funções (mínimo, máximo, logaritmo e exponencial).

ARAÚJO (2009) desenvolveu uma extensão da ferramenta *Illium* para viabilizar a observação da evolução de software através de um modelo baseado nas Leis de Evolução de Software. Nesse trabalho, uma nova versão foi desenvolvida visando melhorar o apoio ao procedimento de simulação, através da inclusão de novas funcionalidades.

2.5 Modelo de Observação de Evolução de Software

O trabalho de ARAÚJO (2009) apresenta uma infraestrutura baseada nas Leis de Evolução de Software (LEHMAN 1980), para observar a evolução do software, mais especificamente, observar o decaimento da qualidade do software através de processos de desenvolvimento e manutenção e assim proporcionar uma maior compreensão de como o software pode ser afetado pelas diferentes mudanças sofridas ao longo do seu ciclo de vida. Para se observar o comportamento de sistemas em evolução, modelos lógicos foram definidos com base em evidências extraídas da literatura técnica de Engenharia de Software, que permitem a simulação de sucessivos ciclos de manutenção, baseado em técnicas de Dinâmica de Sistemas.

A construção do modelo se deu inicialmente com a seleção de características de software que pudessem representar aspectos de qualidade de sistemas e a construção de um conjunto de premissas descritas através de formulações lógicas para cada Lei de Evolução de Software a partir de determinadas características de software, medidas através de métricas apropriadas para cada característica nas diferentes etapas de desenvolvimento do produto (ARAÚJO 2009).

As formulações lógicas construídas para as Leis de Evolução de Software não estabelecem uma relação entre essas Leis e os valores absolutos de suas características, mas sim a descrição de seus comportamentos através das tendências dessas características (aumento, diminuição, constância) (ARAÚJO 2009).

Com o objetivo de investigar as influências entre as características de software selecionadas e proporcionar uma maior validade à construção do modelo, foi realizado um conjunto de estudos baseados em revisão sistemática, onde cada um dos estudos considerava um par das características de software determinadas. Além da influência entre as características, também foi analisado a direção da influência entre as duas características de software, revelando qual das duas características exerce a influência e, a taxa ou intensidade que uma característica de software influencia a outra (ARAÚJO 2009).

2.5.1 Características de Software

Para observar o comportamento da evolução de sistemas de software através das LES, foram selecionadas características de software que pudessem representar os aspectos considerados pelas Leis de Evolução de Software (ARAÚJO 2009). A seleção das características foi feita com base na norma ISO 9126-1 (1997) e na interpretação de cada lei, onde cada característica selecionada pudesse representar aspectos de qualidade de sistemas de software no sentido de proporcionar um melhor entendimento do que realmente afetaria o decaimento de software.

A seguir são apresentadas as características de software selecionadas e suas interpretações, conforme ARAÚJO *et al.* (2005):

Tamanho: caracterizado pela quantidade de artefatos produzidos em cada etapa do ciclo de vida do software proposto;

Periodicidade: representa o intervalo de tempo decorrido entre cada versão produzida de um artefato;

Complexidade: identificada através de elementos que podem medir a complexidade estrutural de um artefato;

Esforço: considerada como o montante de trabalho realizado, medido em termos de homens/hora ou unidade equivalente;

Confiabilidade: representada pela quantidade de defeitos identificados por artefato em cada versão do software, e;

Manutenibilidade: caracterizada pelo tempo gasto na identificação de defeitos e também no tempo gasto com a sua remoção, por versão do artefato.

A Tabela 2-4 abaixo exibe esse conjunto de características de software, indicando seus relacionamentos com as Leis de Evolução de Software.

Tabela 2-4 – Relacionamento entre Leis de Evolução de Software e características de software (adaptado de ARAÚJO, 2009).

	Tamanho	Periodicidade	Complexidade	Esforço	Confiabilidade	Manutenibilidade
Mudança Contínua		✓		✓		
Incremento da Complexidade	✓		✓	✓		✓
Autorregulação	✓				✓	
Conservação da Estabilidade Organizacional				✓		
Conservação da Familiaridade	✓		✓	✓		
Crescimento Contínuo	✓	✓				
Declínio da Qualidade			✓	✓	✓	✓
Sistema de Realimentação	✓	✓	✓	✓	✓	✓

2.5.2 Formulações Lógicas

Após a seleção das características, foi realizado um estudo com o objetivo de compreender como elas se comportariam para levar ao decaimento de software. Para cada Lei de Evolução de Software, foram descritas tabelas verdade no sentido de descrever o comportamento esperado de cada uma das características, considerando as implicações neutra, positiva e negativa de cada Lei de Evolução (ARAÚJO 2009).

A implicação neutra revela uma situação inerente à própria Lei de Evolução de Software. A implicação positiva revela que a Lei pode resultar em uma situação oposta ao decaimento de software, como o uso de técnicas antirregressivas, por exemplo. É importante observar que a implicação positiva não necessariamente resulta em decaimento de software. A implicação negativa é aquela que resulta em decaimento de software. As premissas apresentadas são baseadas nas implicações negativas para cada Lei de Evolução de Software já que este é objetivo do estudo (ARAÚJO 2009). A Tabela 2-5 apresenta as premissas obtidas através de tabelas verdade.

Tabela 2-5 – Premissas descritas para cada Lei de Evolução de Software (adaptado de ARAÚJO 2009).

Lei de Evolução de Software	Premissas
I – Mudança Contínua	(Periodicidade não aumenta Λ Esforço não diminui) → Mudança Contínua
II – Incremento da Complexidade	(Tamanho aumenta V Complexidade aumenta V Esforço aumenta V Manutenibilidade diminui) → Incremento da Complexidade
III – Autorregulação	(Tamanho não aumenta Λ Confiabilidade não diminui Λ Esforço não diminui) → Autorregulação
IV – Conservação da Estabilidade Organizacional	(Esforço não modifica) → Conservação da Estabilidade Organizacional
V – Conservação da Familiaridade	(Tamanho não modifica Λ Complexidade não modifica Λ Esforço não modifica) → Conservação da Familiaridade
VI – Crescimento Contínuo	(Tamanho aumenta Λ Periodicidade não aumenta) → Crescimento Contínuo
VII – Declínio da Qualidade	(Complexidade aumenta V Esforço aumenta V Confiabilidade diminui V Manutenibilidade diminui) → Declínio da Qualidade
VIII – Sistema de Realimentação	(Coleta das medidas relativas à Tamanho, Periodicidade, Complexidade, Esforço, Confiabilidade, Manutenibilidade) → Sistema de Realimentação

Tendo em vista que as formulações lógicas construídas para as LSE não estabelecem uma relação entre essas Leis, foi realizado um conjunto de estudos baseados em revisão sistemática com o objetivo de investigar as influências entre todas as características de software (ARAÚJO 2009).

A Tabela 2-6 apresenta a interpretação geral das implicações neutras, positivas e negativas para cada lei de evolução de software de acordo com ARAÚJO (2009).

Tabela 2-6 – Interpretação das implicações neutras, positivas e negativas (adaptado de ARAÚJO 2009).

Lei de Evolução de Software	Neutra	Positiva	Negativa
I – Mudança Contínua	O sistema estará sujeito à mudança contínua	Se a mudança é bem executada, o sistema permanecerá útil	Se a mudança não é bem executada ou ela não é realizada, o sistema se tornará progressivamente menos útil
II – Incremento da Complexidade	O sistema que é modificado se torna mais complexo	Incremento na complexidade pode ser necessário para refletir modificações no ambiente do sistema e incremento de novos requisitos	Incremento na complexidade pode dificultar a manutenção e a utilização do produto
III – Autorregulação	Sistemas possuem mecanismos de autorregulação	A autorregulação permite gerenciar o processo de evolução de sistemas, permitindo que um equilíbrio seja obtido e explorado para seu gerenciamento	A autorregulação não considerada no gerenciamento do desenvolvimento permite que eventos externos possam influenciar nesses mecanismos de autorregulação
IV – Conservação da Estabilidade Organizacional	Sistemas possuem taxas de atividade global ao longo do ciclo de vida	As taxas de atividade global tendem a permanecer constantes ao longo do ciclo de vida, indicando estabilidade no desenvolvimento do sistema	As taxas de atividade global ao longo do ciclo de vida variam significativamente, evidenciando uma instabilidade no desenvolvimento do sistema.
V – Conservação da Familiaridade	Sistemas possuem taxas de crescimento médio constantes	As taxas de crescimento médio tendem a diminuir, evidenciando estabilidade no desenvolvimento do sistema	As taxas de crescimento médio tendem a aumentar ou manter-se constantes, indicando possibilidade de esforço excessivo entre os ciclos de evolução do sistema, evidenciando instabilidade no desenvolvimento do sistema
VI – Crescimento Contínuo	Sistemas em uso estão sujeitos ao crescimento contínuo	O incremento de novas funcionalidades pode manter a utilidade do produto, permanecendo útil em relação às mudanças no ambiente e às expectativas dos usuários	A falta de incremento de novas funcionalidades pode tornar o sistema progressivamente menos útil em relação às necessidades dos usuários
VII – Declínio da Qualidade	Um produto em evolução tende a declinar em qualidade	Declínio da qualidade pode promover práticas antirregressivas, como redocumentação, reestruturação, reengenharia, engenharia reversa	Declínio da qualidade pode tornar o entendimento e a modificação do produto mais difícil
VIII – Sistema de Realimentação	Sistemas em evolução possuem mecanismos de realimentação	O sistema de realimentação é tratado de forma a alcançar melhorias significativas na qualidade de um sistema	O sistema de realimentação não é considerado, ou não é utilizado, no sentido de alcançar melhorias significativas na qualidade de um sistema

2.5.3 Visão Geral do Protocolo da quasi Revisão Sistemática

A construção do modelo de observação de evolução de software original (ARAÚJO 2009) foi realizada com base na organização de um protocolo de pesquisa com objetivo de encontrar evidências sobre os relacionamentos entre as características de software selecionadas, analisando a direção e a taxa/intensidade da influência.

Os relacionamentos entre as características também foram analisados considerando as diferentes etapas do desenvolvimento de software orientado a objetos, com base nas etapas propostas por TRAVASSOS *et al.* (2001): Especificação de Requisitos, Projeto de Alto Nível, Projeto de Baixo Nível e Codificação, comumente encontradas em processos de desenvolvimento de software. De acordo com ARAÚJO (2009), a etapa de Testes não é considerada uma vez que a infraestrutura proposta objetiva a observação do processo de modificação de artefatos de software e não sua verificação, validação ou teste.

Portanto, a construção do modelo consistiu em analisar 15 questões principais de pesquisa, executadas no final de 2006. As questões de pesquisa definidas no protocolo da *quasi* revisão sistemática (ARAÚJO 2009) tiveram o objetivo de avaliar a existência do relacionamento, a direção e a taxa de influência entre as seguintes características:

Q01: Tamanho e Complexidade
Q02: Tamanho e Confiabilidade
Q03: Complexidade e Esforço
Q04: Esforço e Confiabilidade
Q05: Complexidade e Manutenibilidade
Q06: Esforço e Manutenibilidade
Q07: Esforço e Periodicidade
Q08: Periodicidade e Manutenibilidade
Q09: Tamanho e Esforço
Q10: Tamanho e Manutenibilidade
Q11: Periodicidade e Tamanho
Q12: Complexidade e Confiabilidade
Q13: Periodicidade e Complexidade
Q14: Manutenibilidade e Confiabilidade
Q15: Periodicidade e Confiabilidade

Para investigar as 15 questões de pesquisa, considerando cada etapa do processo de desenvolvimento de software, foi elaborado um meta-protocolo para a *quasi* revisão sistemática (ARAÚJO 2009). Para cada par de características, quatro *strings* de busca foram consideradas, uma para cada etapa de desenvolvimento analisada. Cada *string* aborda 3 sub-questões de pesquisa. A primeira sub-questão considera a existência do relacionamento entre o par de características. A segunda sub-questão considera a direção do relacionamento e a terceira a taxa/intensidade da influência entre as características. Por exemplo, para o par Q9, considerando a etapa de Codificação, as questões são apresentadas como: Existe influência entre as características de software Tamanho e Esforço na etapa de Codificação de um processo de desenvolvimento de software orientado a objetos? Qual a direção da influência entre as características de software Tamanho e Esforço na etapa de Codificação de um processo de desenvolvimento de software orientado a objetos? Qual a intensidade/taxa da influência entre as características de software Tamanho e Complexidade na etapa de Codificação de um processo de desenvolvimento de software orientado a objetos?

A fonte de informação para a pesquisa foi representada por algumas bibliotecas digitais que indexam publicações de conferências e *journals* na área de Engenharia de Software. Adicionalmente, foram considerados *proceedings* de conferências em que o tema se relacionava com evolução ou manutenção de software.

A biblioteca digital *EI Compendex* foi utilizada como a máquina de busca para executar as *strings.* Entre as fontes indexadas por essa máquina, estão *IEEE* e *ACM*. A busca compreendeu todos os artigos publicados até o final de 2006.

As *strings* de busca foram construídas com base nas seguintes palavras-chave: *Software Characteristic, metric, relation, relationship, correlation, dependency, influence, effect, direction, primary study, experimental study, empirical study, intensity and rate.* Além de palavras-chave específicas para cada característica e etapa de desenvolvimento em questão.

Para a seleção dos artigos retornados pela máquina de busca foram definidos critérios de inclusão e exclusão. Os critérios consideram artigos teóricos, provas de conceito, estudos experimentais, escritos em língua inglesa, disponíveis na Internet e que apresentem estudos sobre o relacionamento entre as características de software analisadas ou suas respectivas métricas. Além disso, os artigos devem considerar estudos sobre sistemas orientados a objeto definidos como *E-Type Systems*

(PFLEEGER 1998) e também sobre a direção e a intensidade/taxa dos relacionamentos entre as características de software analisadas.

A seleção de artigos foi realizada por dois pesquisadores e consistiu em duas análises. A primeira consistiu na leitura do título, resumo e conclusões. A segunda e mais detalhada análise consistiu na leitura integral do artigo. Os critérios de inclusão e exclusão foram aplicados em ambas.

Para registrar as informações necessárias para responder as questões de pesquisa e também para dar suporte à avaliação dos artigos selecionados, foram definidas diretrizes para a extração de informações. As seguintes informações foram extraídas: título, autor, fonte, data de publicação, tipo do estudo, categoria, contexto, tecnologias utilizadas, estágios de desenvolvimento envolvidos, características de software e métricas utilizadas e a descrição da influência (direção e intensidade/taxa) entre as características de software analisadas.

Um processo de avaliação da qualidade dos artigos foi definido para selecionar aqueles que estavam de acordo com o propósito deste estudo**.** Para avaliar a qualidade os artigos selecionados foram definidos oito itens como critérios para avaliação e eles estão relacionados à análise de dados utilizados no estudo (identificação e tratamento de *outliers*), a aplicação de métodos de análise de sensibilidade ou residual, o uso apropriado de métodos estatísticos, a apresentação de informações sobre os projetos utilizados, os métodos de comparação aplicados, a acurácia dos resultados e a apresentação da metodologia utilizada para alcançar os resultados. Entre os critérios utilizados, aqueles relacionados ao tratamento dos dados e aos métodos estatísticos utilizados foram considerados como mais importantes.

2.5.4 O Modelo Conceitual de Observação de Evolução de Software

Com base nas evidências encontradas na revisão da literatura técnica de Engenharia de Software, a Figura 2-3 apresenta o Diagrama de Causas e Efeitos que descreve o modelo de observação de evolução de software proposto por ARAÚJO (2009).

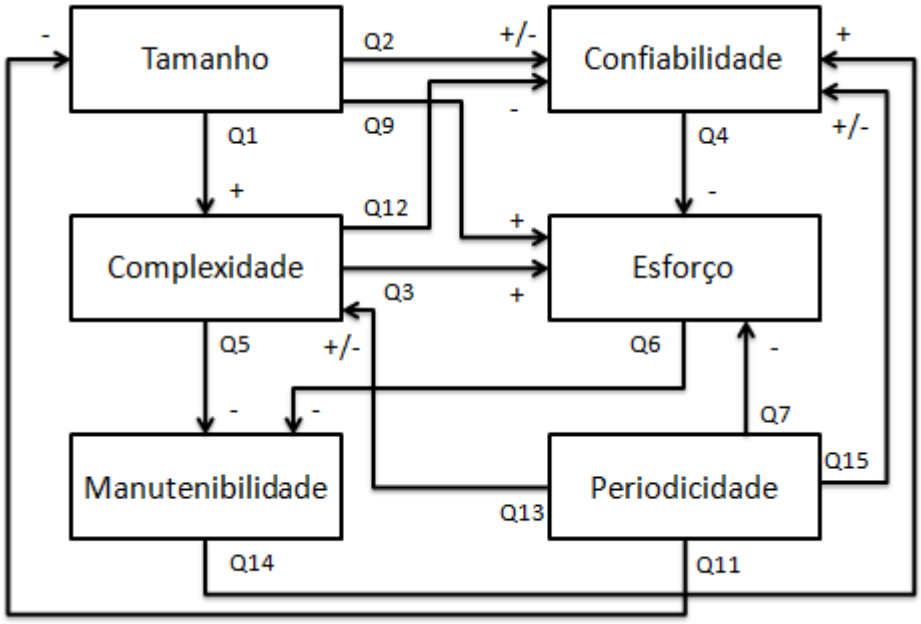


Figura 2-3 – Diagrama de Causas e Efeitos relacionado às características de software (adaptado de ARAÚJO 2009).

Na Figura 2-3, a identificação de cada ligação é apresentada por Qx, onde x é um número entre 1 e 15 e representa a questão de pesquisa correspondente. Segundo a notação do Diagrama de Causas e Efeitos, uma ligação entre duas características de software indica uma relação de influência entre elas; as setas existentes nessas ligações indicam a direção da influência entre as características de software e os sinais associados a cada relacionamento indicam a tendência de aumento ou diminuição de uma característica justificada pelos indícios encontrados na literatura técnica a partir das revisões realizadas para cada par de características (ARAÚJO 2009). Por exemplo, a característica Complexidade influencia positivamente a característica Esforço, isto é, quanto maior a Complexidade (medida através da complexidade ciclomática, da profundidade de herança ou de outras métricas), maior será o Esforço no desenvolvimento ou manutenção do sistema.

2.5.5 O Modelo de Dinâmica de Sistemas

Através da identificação das ligações e influências entre as características de software, foi construído um modelo dinâmico, no sentido de simular o comportamento de sistemas de software orientados a objetos para observar o processo de evolução, mais especificamente de decaimento, através de sucessivos ciclos de manutenção (ARAÚJO 2009).

Para a construção do modelo de Dinâmica de Sistemas, ARAÚJO (2009) utilizou a linguagem de modelagem definida pela ferramenta de simulação *Illium* (BARROS *et al.* 2000).

As equações contidas no modelo são construídas através de regressão linear, aplicando o método de mínimos quadrados para identificar a tendência de comportamento daquela característica de software. A regressão linear, apesar da possibilidade de aumento do erro, é condizente com a análise semiquantitativa de dados, em que a tendência do comportamento de uma variável é o que deve ser considerado, mais do que seus valores individuais (ARAÚJO 2009).

No modelo apresentado algumas características são influenciadas por mais de uma característica, por exemplo, a característica Complexidade é influenciada pelas características Tamanho e Periodicidade. A equação para a característica Complexidade aparentemente está somando variáveis com unidades diferentes, entretanto, é importante ressaltar que a construção da equação foi feita através de regressão linear, o que elimina a diferença entre as unidades das variáveis que estão sendo somadas. O modelo é apresentado a seguir.

```
#
######################################################################
SPEC DT 1.00;
######################################################################
#
# SOFTWARE CHARACTERISTICS
#
######################################################################
STOCK SIZE 63.29;
STOCK PERIODICITY 4.00;
STOCK COMPLEXITY 11.43;
STOCK EFFORT 0.50;
STOCK RELIABILITY 1.00;
STOCK MAINTAINABILITY 0.50;
STOCK TIMER 1.00;
######################################################################
#
# RATES
#
######################################################################
```

PROC DELTA_PERIODICITY 0.23 * TIMER;
PROC DELTA_SIZE 0.026 * DELTA_PERIODICITY;
PROC DELTA_COMPLEXITY 0.07 * DELTA_SIZE + 0.01 * DELTA_PERIODICITY;
PROC DELTA_MAINTAINABILITY -11.95 * DELTA_COMPLEXITY;
PROC DELTA_RELIABILITY -0.92 * DELTA_SIZE + 0.20 *
DELTA_PERIODICITY + 2.23 * DELTA_COMPLEXITY + 0.17 *
DELTA_MAINTAINABILITY;
PROC DELTA_EFFORT -8.18 * DELTA_SIZE + 0.39 * DELTA_PERIODICITY + (-75.36 * DELTA_COMPLEXITY) + 4.55 * DELTA_RELIABILITY;

RATE (**SOURCE**, TIMER) RTT DT;
RATE (**SOURCE**, PERIODICITY) RTP **MAX** (0, DELTA_PERIODICITY);
RATE (**SOURCE**, SIZE) RTS MAX (0, DELTA_SIZE);
RATE (**SOURCE**, COMPLEXITY) RTC **MAX** (0, DELTA_COMPLEXITY);
RATE (**SOURCE**, MAINTAINABILITY) RTM **MAX** (0, DELTA_MAINTAINABILITY);
RATE (**SOURCE**, RELIABILITY) RTR **MAX** (0, DELTA_RELIABILITY);
RATE (**SOURCE**, EFFORT) RTE **MAX** (0, DELTA_EFFORT);

De acordo com ARAÚJO (2009), a constante **DT**, definida com o valor 1, estabelece o incremento de tempo (em dias) utilizado no processo de simulação. Um **STOCK** é definido para cada uma das características de software. O valor definido para cada um representa o valor inicial da variável e este valor é obtido a partir da última versão do sistema em observação, para cada característica. O **STOCK** TIMER armazena o acúmulo de iterações desde o início do processo de simulação. As equações que representam os relacionamentos entre as características de software são definidas como um **PROC.** Observa-se a utilização de variáveis para cada característica prefixadas por DELTA, evidenciando a variação ocorrida para aquela característica em uma determinada iteração no processo de simulação. Por fim, é definida uma **RATE** para cada equação, que efetivamente faz a transferência dos valores para os repositórios, em função da taxa definida em cada uma das equações. Cada **RATE** foi construída de forma a não permitir que as características de software atinjam valores negativos, os quais não fariam sentido para as características consideradas. Observa-se ainda que cada **RATE** utiliza como primeiro parâmetro a palavra reservada **SOURCE**, que identifica uma fonte infinita do elemento representado no **STOCK.**

2.5.6 *Illium Software Evolution*

Para apoiar a execução do modelo construído com base em Dinâmica de Sistemas, foi desenvolvida uma extensão da ferramenta *Illium* (BARROS *et al.* 2000). Essa ferramenta foi desenvolvida de forma a apoiar a simulação de análise de riscos em projetos de software. Sua capacidade de expansão permitiu a implementação das funcionalidades necessárias com relação à geração das equações para o processo de

simulação e a interpretação das Leis de Evolução de Software a partir do modelo proposto. A versão estendida da ferramenta foi chamada de *Illium Software Evolution* (ARAÚJO 2009).

A extensão desenvolvida é apenas uma interface gráfica que utiliza a máquina de simulação de *Illium* para executar a simulação e também utiliza os resultados gerados para calcular a situação de cada Lei de Evolução de Software de acordo com as premissas apresentadas.

A utilização da ferramenta proposta compreende um conjunto de atividades, descritas a seguir. Inicialmente, o Engenheiro de Software coleta dados de um sistema real, no qual deseja observar o processo de evolução. É importante que as medidas coletadas sejam homogêneas para um determinado sistema, ou seja, estejam sempre na mesma unidade para uma determinada característica de software. As medidas coletadas, referentes a cada uma das características de software analisadas, são armazenadas em uma planilha Excel fornecida junto com a ferramenta. Na planilha, os dados são utilizados para gerar as equações utilizadas no processo de simulação, através de regressão linear, em que a tendência é o foco, mais do que a precisão de valores reais (ARAÚJO 2009).

O conjunto de equações geradas pode ser computado através da ferramenta *Illium Software Evolution*, simulando o comportamento de sistemas de software em evolução. Por fim, através dessa ferramenta, pode-se observar o resultado da simulação e o processo de decaimento de software, indicando-se ainda o estado de cada Lei de Evolução de Software em função das tendências de comportamento das características de software e do modelo considerado (ARAÚJO 2009).

Como se pode observar, o conjunto de atividades descritas para a construção do modelo de Dinâmica de Sistemas e para execução da simulação representa um processo semi-automatizado, já que envolve a utilização de uma planilha eletrônica para a construção do modelo pelo Engenheiro de Software.

O modelo de observação de evolução de software foi originalmente apresentado como um modelo para observar a evolução de um software nas diferentes etapas do desenvolvimento: Projeto de Alto Nível, Projeto de Baixo Nível, Especificação de Requisitos e Codificação (ARAÚJO 2009). Entretanto, o procedimento de simulação deve ser executado separadamente para cada etapa. Para isso, foi sugerido um conjunto de possíveis métricas, para cada uma das características de software utilizadas no modelo, considerando cada uma das etapas do desenvolvimento. O conjunto de métricas definido por ARAÚJO (2009) é apresentado na Tabela 2-7.

Tabela 2-7 – Métricas associadas por Característica em cada Etapa de Desenvolvimento de Software (adaptado de ARAÚJO 2009).

	Tamanho	Periodicidade	Complexidade	Esforço	Confiabilidade	Manutenibilidade
Especificação de Requisitos	• Número de Pontos de Função • Número de Pontos de Casos de Uso • Número de Requisitos	• Intervalo entre Versões	• Número de Casos de Uso	• Número de Requisitos Tratados • Número de Casos de Uso Tratados • Número de Pessoas • Recursos Alocados • Tempo Consumido • Produtividade Média da Equipe	• Número de Defeitos Detectados • Número de Defeitos Corrigidos	• Tempo Consumido no Diagnóstico de Defeitos • Tempo Consumido na Remoção de Defeitos
Projeto de Alto Nível	• Número de Classes • Número de Métodos por Classe	• Intervalo entre Versões	• Número de Diagramas de Classes • Número de Diagramas de Sequência • Número de Diagramas de Estado • Número de Diagramas de Empacotamento • Número de Diagramas de Atividades • Profundidade de Herança por Classe • Número de Filhos por Classe	• Número de Diagramas de Classes Tratados • Número de Diagramas de Sequência Tratados • Número de Diagramas de Estado Tratados • Número de Diagramas de Empacotamento Tratados • Número de Diagramas de Atividades Tratados • Número de Pessoas • Recursos Alocados • Tempo Consumido • Produtividade Média da Equipe	• Número de Defeitos Detectados • Número de Defeitos Corrigidos	• Tempo Consumido no Diagnóstico de Defeitos • Tempo Consumido na Remoção de Defeitos
Projeto de Baixo Nível	• Número de Classes de Domínio • Número de Classes de Suporte • Número de Métodos por Classe • Número de Subsistemas	• Intervalo entre Versões	• Número de Diagramas de Classes • Número de Diagramas de Sequência • Profundidade de Herança por Classe • Acoplamento entre Objetos • Resposta de uma Classe • Falta de Coesão em Métodos • Número de Filhos por Classe	• Número de Diagramas de Classes Tratados • Número de Diagramas de Sequência Tratados • Número de Pessoas • Recursos Alocados • Tempo Consumido • Produtividade Média da Equipe	• Número de Defeitos Detectados • Número de Defeitos Corrigidos	• Tempo Consumido no Diagnóstico de Defeitos • Tempo Consumido na Remoção de Defeitos
Codificação	• Número de Linhas de Código Fonte • Número de Métodos por Classe	• Intervalo entre Versões	• Profundidade de Herança por Classe • Acoplamento entre Objetos • Resposta de uma Classe • Falta de Coesão em Métodos • Número de Filhos por Classe • Complexidade Ciclomática por Método	• Número de Linhas de Código Fonte Tratados • Número de Pessoas • Recursos Alocados • Tempo Consumido • Produtividade Média da Equipe	• Número de Defeitos Detectados • Número de Defeitos Corrigidos • Disponibilidade do Sistema	• Tempo Consumido no Diagnóstico de Defeitos • Tempo Consumido na Remoção de Defeitos

2.6 Conclusão

Neste capítulo foram apresentados conceitos relacionados à evolução de software, simulação e o modelo de evolução de observação de software previamente proposto, que é o objeto de estudo deste trabalho.

No próximo capítulo serão apresentados os resultados da *quasi* revisão sistemática executada no contexto deste trabalho e a atualização do modelo conceitual original. Além disso, o novo modelo conceitual será apresentado.

3 Atualização do Modelo de Observação de Evolução de Software

Nesse capítulo serão apresentadas as evidências adicionais encontradas na literatura técnica de Engenharia de Software através da atualização e evolução de um estudo baseado em revisão sistemática. O estudo analisa os relacionamentos entre as características de software utilizadas como base para construir o modelo de observação de evolução de software proposto originalmente por ARAÚJO (2009).

As evidências adicionais encontradas foram utilizadas para realizar uma atualização do modelo original. A atualização ocorreu sob dois aspectos distintos: de alteração e reafirmação das relações apresentadas no modelo. A alteração do modelo ocorreu devido às evidências adicionais encontradas, que sugeriram a inclusão da relação entre duas características de software utilizadas no modelo original. A reafirmação do modelo ocorreu devido às evidências adicionais encontradas, que reforçam a existência de relacionamento entre as características de software utilizadas no modelo.

3.1 Revisão Sistemática

Geralmente, as revisões da literatura técnica são executadas através de métodos subjetivos e informais para coletar e interpretar estudos e os pesquisadores tendem a citar seletivamente estudos que reforçam preconceitos, ou seja, que seguem uma mesma corrente de raciocínio. Por outro lado, a revisão sistemática se preocupa em realizar uma pesquisa abrangente, compreensiva e exaustiva para encontrar estudos relacionados a uma questão de pesquisa formalmente definida. Além disso, utiliza critérios bem definidos para seleção de estudos, para a avaliação da qualidade dos estudos e também define uma síntese dos resultados de acordo com métodos predeterminados (PAI *et al.* 2004).

A definição de um protocolo de pesquisa contendo informações para a execução e apresentação dos resultados do estudo secundário baseado em revisão sistemática, tais como, definição das questões de pesquisa, critérios de inclusão e exclusão para a seleção de artigos, critérios para avaliação dos resultados apresentados pelos artigos, entre outras informações, confere um alto grau de formalidade a este tipo de estudo. Essa formalidade traz grandes benefícios para a comunidade envolvida com o assunto alvo do estudo secundário. Dentre eles, a reutilização dos resultados como referência para outras pesquisas, a possibilidade de

realizar uma re-execução do estudo, para uma possível avaliação ou atualização dos resultados obtidos e, além disso, confere maior validade conceitual ao estudo (PAI *et al.* 2004).

Devido à formalidade utilizada nos estudos baseados em revisão sistemática, neste capítulo serão apresentados os resultados da evolução e atualização do estudo realizado por ARAÚJO (2009). De acordo com TRAVASSOS *et al.* (2008), o conceito de *quasi* revisão sistemática representa um tipo de revisão sistemática que utiliza o mesmo rigor e formalismo para as fases metodológicas de elaboração e execução do protocolo de pesquisa, onde a estrutura PICO (PAI *et al.* 2004) tem o campo *Comparison* representado por um conjunto vazio, reduzindo assim a possibilidade de executar meta-análise ou qualquer tipo de síntese de pesquisa mais elaborada.

3.2 Evolução do Protocolo de quasi Revisão Sistemática

O estudo secundário de *quasi* revisão sistemática executado no contexto deste trabalho representa uma evolução do estudo realizado por ARAÚJO (2009). Isso se deve ao fato de ter sido realizada uma atualização do estudo e, além disso, a inclusão de uma nova base digital de artigos como fonte de pesquisa com o objetivo de aumentar a abrangência das buscas e melhorar os resultados.

A atualização do estudo de *quasi* revisão sistemática consistiu na re-execução do protocolo de pesquisa definido em ARAÚJO (2009). Para isso, o protocolo foi utilizado sem qualquer alteração em sua estrutura geral, com o objetivo de manter a coerência entre os resultados obtidos na primeira execução com os resultados obtidos nesta segunda execução do estudo. Portanto, a atualização do estudo consistiu em utilizar o mesmo protocolo de *quasi* revisão sistemática alterando apenas o parâmetro de pesquisa relacionado à data de publicação dos artigos. Na atualização do estudo, a nova data considerada para a máquina de busca *EI Compendex*, usada no protocolo original, compreende o período entre 2007 e 2010, já que anteriormente se considerou os artigos publicados até o final de 2006. Dentre as fontes de publicação indexadas pela *EI Compendex* estão *IEEE* e *ACM.*

Uma importante modificação realizada no protocolo, que não altera sua estrutura geral, foi o acréscimo de mais uma fonte de pesquisa, com o objetivo de ampliar a abrangência do estudo para dar maior confiabilidade aos resultados obtidos. A nova fonte de pesquisa incluída foi a máquina de busca *Scopus*, que indexa os artigos das principais fontes de publicação da área de Engenharia de Software, dentre elas, *IEEE, ACM, Cite Seerx, Elsevier e Springer*. Para a nova máquina de busca

incluída ao protocolo foi necessário realizar a busca sem restrição de data, ou seja, buscar em artigos publicados até a data de execução deste estudo, outubro de 2010.

3.3 Resultados da quasi Revisão Sistemática

Em cada uma das máquinas de busca, *Scopus* e *EI Compendex*, foram executadas 60 *strings* de busca, já que para cada uma das 15 questões de pesquisa foram criadas 4 *strings* de busca relacionadas com cada etapa do processo de desenvolvimento de software.

A estrutura das *strings* de busca seguiram um padrão que considerava o estágio de desenvolvimento, as características de software analisadas e as questões de pesquisa definidas no protocolo elaborado por ARAÚJO (2009). Devido a este padrão, algumas das palavras-chave se repetiam nas diferentes *strings*, o que afetou os resultados com relação ao número de artigos duplicados entre os artigos retornados por cada busca. O padrão da estrutura das *strings* de busca é apresentado a seguir.

(Palavras-chave relacionadas às questões de pesquisa)
AND
(Palavras-chave relacionadas a uma das características de software)
AND
(Palavras-chave relacionadas a outra característica de software)
AND
(Palavras-chave relacionadas à etapa do processo de desenvolvimento de software)
AND
(Palavras-chave relacionadas à população a ser analisada)

Para exemplificar, a questão de pesquisa para a questão Q1 (Influência Tamanho e Complexidade), considerando a etapa de Especificação de Requisitos, é apresentada a seguir.

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage)
AND
(Size OR {Function Points} OR {Use Case Points} OR {Requirement})
AND

(Complexity OR {use case})
AND
({Requirement Specification} OR {Requirement Elicitation} OR {Requirement Definition} OR Analysis OR {User Requirement} OR {Requisite})
AND
({Software Characteristic} OR {Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

O total de artigos retornados para as 60 *strings* de busca, considerando as duas máquinas de busca utilizadas, *Scopus* e *EI Compendex,* foi de 6753, onde a maioria eram artigos duplicados. Inicialmente esses artigos foram agrupados por questão de pesquisa e mesmo entre diferentes questões existiam artigos duplicados, já que uma mesma característica estava presente em diferentes questões de pesquisa e alguns artigos tratavam de várias das características analisadas. Para esses casos não foi possível fazer uma separação de acordo com a questão de pesquisa. Por exemplo, se um artigo que foi retornado para as questões Q1 e Q2, no primeiro momento, ele só será contabilizado para a questão Q1. Entretanto, após a extração e análise do conteúdo do artigo ele será considerado para as duas questões caso seja pertinente. Para apoiar à organização, classificação e análise dos artigos retornados, a ferramenta JabRef (JABREF 2011) foi utilizada. Após a eliminação dos artigos duplicados restaram 884 artigos. A Tabela 3-1 mostra a quantidade de artigos retornados para cada questão não considerando os artigos duplicados.

Tabela 3-1 – Total de artigos retornados eliminando as duplicatas.

Questão – Características Relacionadas	**# Artigos Retornados sem Duplicatas**	
	Scopus (até 2010)	**EI Compendex (2007-2010)**
Q1 – Tamanho e Complexidade	265	19
Q2 – Tamanho e Confiabilidade	90	6
Q3 – Complexidade e Esforço	104	13
Q4 – Confiabilidade e Esforço	59	5
Q5 – Complexidade e Manutenibilidade	38	3
Q6 – Esforço e Manutenibilidade	37	11
Q7 – Periodicidade e Esforço	0	1
Q8 – Periodicidade e Manutenibilidade	0	0
Q9 – Tamanho e Esforço	155	11
Q10 – Tamanho e Manutenibilidade	21	2
Q11 – Periodicidade e Tamanho	0	1
Q12 – Complexidade e Confiabilidade	17	5
Q13 – Periodicidade e Complexidade	0	2
Q14 – Manutenibilidade e Confiabilidade	17	2
Q15 – Periodicidade e Confiabilidade	0	0
Total de Artigos	**803**	**81**

Todos os 884 artigos passaram por uma primeira análise executada por dois pesquisadores. O primeiro pesquisador fez a seleção dos artigos, com base nos critérios de inclusão e exclusão definidos no protocolo, através da leitura do título, do resumo e das conclusões dos artigos. O segundo pesquisador, mais experiente e especialista na área, analisou os resultados da seleção realizada para garantir que os artigos incluídos e excluídos estivessem de acordo com o propósito da pesquisa e garantir a qualidade desse processo. Após a primeira análise, foram selecionados 98 artigos, onde 78 foram retornados da *Scopus* e 19 retornados da *EI Compendex.* A Tabela 3-2 mostra a quantidade de artigos por questão após a primeira análise.

Tabela 3-2 – Quantidade de artigos por questão após primeira análise.

Questão – Características Relacionadas	**# Artigos após primeira análise**	
	Scopus (até 2010)	**EI Compendex (2007-2010)**
Q1 – Tamanho e Complexidade	30	3
Q2 – Tamanho e Confiabilidade	8	3
Q3 – Complexidade e Esforço	7	3
Q4 – Confiabilidade e Esforço	1	1
Q5 – Complexidade e Manutenibilidade	5	0
Q6 – Esforço e Manutenibilidade	4	1
Q7 – Periodicidade e Esforço	0	0
Q8 – Periodicidade e Manutenibilidade	0	0
Q9 – Tamanho e Esforço	17	7
Q10 – Tamanho e Manutenibilidade	1	0
Q11 – Periodicidade e Tamanho	0	0
Q12 – Complexidade e Confiabilidade	6	1
Q13 – Periodicidade e Complexidade	0	0
Q14 – Manutenibilidade e Confiabilidade	0	0
Q15 – Periodicidade e Confiabilidade	0	0
Total de Artigos	**79**	**19**

Os 98 artigos que passaram da primeira análise foram submetidos a uma segunda análise mais detalhada. Essa análise consistiu em uma leitura completa do artigo visando verificar os critérios de inclusão e exclusão definidos no protocolo de *quasi* revisão sistemática definido por ARAÚJO (2009). Novamente, essa análise foi realizada por dois pesquisadores visando garantir melhores resultados para o estudo. O primeiro pesquisador realizou a leitura completa dos artigos e selecionou aqueles que estavam de acordo com os critérios de inclusão e exclusão. O segundo pesquisador, mais experiente, analisou os resultados dessa seleção para garantir que as inclusões e exclusões foram estavam de acordo com o propósito da pesquisa. Mais detalhes sobre a reexecução do estudo baseado em *quasi* revisão sistemática estão disponíveis no Apêndice A.

Após a leitura completa dos 98 artigos, 24 foram selecionados da máquina de busca *Scopus* e 7 foram selecionados da máquina de busca *EI Compendex*. Dentre os

31 artigos selecionados, 4 deles foram retornados por ambas as máquinas de busca e foram desconsiderados, já que eram duplicatas.

Portanto, 27 artigos foram selecionados como possíveis evidências para este estudo. Em seguida, os artigos selecionados passaram por um processo de extração de informações cujo objetivo foi organizar e identificar aqueles que realmente possuíam evidências para o estudo. Nesse processo, as seguintes informações foram identificadas e extraídas dos artigos: questões de pesquisa tratadas no artigo, título, ano de publicação, autor, fonte, contexto, métricas e características de softwares analisadas e a descrição, direção e intensidade da influência entre as métricas ou características de software.

A extração de informações permitiu que fossem identificados artigos que apresentavam evidências para mais de uma questão de pesquisa e assim um mesmo artigo pôde ser considerado para mais de uma questão de pesquisa. A Tabela 3-3 mostra a quantidade de artigos com evidências por questão.

Tabela 3-3 – Quantidade de artigos com evidências por questão após análise detalhada.

Questão – Características Relacionadas	**# Artigos após análise detalhada**		**Total de Artigos por Questão**
	Scopus (até 2010)	**EI Compendex (2007-2010)**	
Q1 – Tamanho e Complexidade	1	1	**2**
Q2 – Tamanho e Confiabilidade	5	0	**5**
Q3 – Complexidade e Esforço	5	2	**7**
Q4 – Confiabilidade e Esforço	0	0	**0**
Q5 – Complexidade e Manutenibilidade	2	0	**2**
Q6 – Esforço e Manutenibilidade	1	0	**1**
Q7 – Periodicidade e Esforço	0	0	**0**
Q8 – Periodicidade e Manutenibilidade	0	0	**0**
Q9 – Tamanho e Esforço	5	3	**8**
Q10 – Tamanho e Manutenibilidade	1	0	**1**
Q11 – Periodicidade e Tamanho	1	0	**1**
Q12 – Complexidade e Confiabilidade	3	1	**4**
Q13 – Periodicidade e Complexidade	0	0	**0**
Q14 – Manutenibilidade e Confiabilidade	0	0	**0**
Q15 – Periodicidade e Confiabilidade	0	0	**0**
Total de Artigos	**24**	**7**	**31**

Após a extração de informações, foi realizada uma avaliação dos artigos com base em critérios de qualidade definidos por ARAÚJO (2009), para selecionar aqueles que estavam de acordo com o propósito dessa pesquisa. Os critérios de avaliação aplicados na seleção dos artigos foram definidos com o objetivo principal de identificar aqueles que aplicaram métodos ou procedimentos estatísticos para fornecer maior qualidade aos resultados apresentados. É importante ressaltar que a qualidade dos artigos foi analisada frente às necessidades dessa pesquisa, e não de uma forma

geral, ou seja, não significa que os artigos desconsiderados tenham baixa qualidade, apenas não atendem aos propósitos da pesquisa.

Após a avaliação da qualidade dos 27 artigos, 5 deles, apesar de apresentarem evidências para o estudo, foram desconsiderados, pois não atendiam aos critérios de avaliação esperados para o contexto dessa pesquisa.

A partir do processo de extração de informações dos artigos, foram encontradas 35 evidências adicionais, extraídas de um total de 22 artigos, para serem utilizadas na atualização do modelo de observação de evolução de software construído por ARAÚJO (2009). Grande parte das evidências adicionais encontradas, isto é, 29 das 35, reafirmaram a existência das relações apresentadas no modelo original, o que confere uma maior confiança ao modelo construído anteriormente. Quatro evidências sugeriram a adição de um novo relacionamento ao modelo e duas sugeriram como alguns dos relacionamentos deveriam ser explorados.

A Tabela 3-4 apresenta a quantidade de evidências adicionais encontradas para cada questão. É importante ressaltar que a restrição para a data de publicação dos artigos da busca na base *Scopus* foi outubro de 2010, data de execução do estudo, e que na *EI Compendex* a restrição para a data compreendia o período entre 2007 e 2010. Isso afetou na quantidade de artigos retornados por cada base.

Tabela 3-4 – Quantidade de evidências adicionais encontradas.

Questão – Características Relacionadas	**# Evidências adicionais**		**Total de Evidências Adicionais por Questão**
	Scopus (até 2010)	**EI Compendex (2007 - 2010)**	
Q1 – Tamanho e Complexidade	1	1	**2**
Q2 – Tamanho e Confiabilidade	4	2	**6**
Q3 – Complexidade e Esforço	4	0	**4**
Q4 – Confiabilidade e Esforço	0	0	**0**
Q5 – Complexidade e Manutenibilidade	5	0	**5**
Q6 – Esforço e Manutenibilidade	1	0	**1**
Q7 – Periodicidade e Esforço	0	0	**0**
Q8 – Periodicidade e Manutenibilidade	0	0	**0**
Q9 – Tamanho e Esforço	6	1	**7**
Q10 – Tamanho e Manutenibilidade	4	0	**4**
Q11 – Periodicidade e Tamanho	0	0	**0**
Q12 – Complexidade e Confiabilidade	4	2	**6**
Q13 – Periodicidade e Complexidade	0	0	**0**
Q14 – Manutenibilidade e Confiabilidade	0	0	**0**
Q15 – Periodicidade e Confiabilidade	0	0	**0**
Total de Evidências	**29**	**7**	**35**

A Tabela 3-5 apresenta o relacionamento entre os artigos selecionados como evidências e as questões de pesquisa, já que alguns artigos apresentaram evidências para mais de uma questão.

Tabela 3-5 – Relação entre os artigos que apresentam evidências para o estudo e as questões de pesquisa.

Questão – Características Relacionadas	Artigos com Evidências Adicionais
Q1 – Tamanho e Complexidade	AGGARWAL ET AL (2006) HEIJSTEK E CHAUDRON (2009)
Q2 – Tamanho e Confiabilidade	El EMAM ET AL (2002) ENGLISH ET AL (2009) FENTON ET AL (2008) HEIJSTEK E CHAUDRON (2009) LESZAK (2005) ZHANG H. B(2009)
Q3 – Complexidade e Esforço	BOCCO ET AL (2005) DARCY ET AL (2005) KOZLOV ET AL (2008) SENTAS ET AL (2008)
Q4 – Confiabilidade e Esforço	Sem evidência adicional.
Q5 – Complexidade e Manutenibilidade	CANFORA ET AL (2005) CRUZ-LEMUS ET AL (2008) KOZLOV ET AL (2008) POELS E DEDENE (2001) RIAZ ET AL (2009)
Q6 – Esforço e Manutenibilidade	RIAZ ET AL (2009)
Q7 – Periodicidade e Esforço	Sem evidência adicional.
Q8 – Periodicidade e Manutenibilidade	Sem evidência adicional.
Q9 – Tamanho e Esforço	DOLADO (2001) FERRUCCI ET AL (2008) HEIJSTEK E CHAUDRON (2009) KOZLOV ET AL (2008) SENTAS ET AL (2008) TSUNODA ET AL (2009) ZHANG J.G. (2008b)
Q10 – Tamanho e Manutenibilidade	CANFORA ET AL (2005) CRUZ-LEMUS ET AL (2008) KOZLOV ET AL (2008) RIAZ ET AL (2009)
Q11 – Periodicidade e Tamanho	Sem evidência adicional.
Q12 – Complexidade e Confiabilidade	AGGARWAL ET AL (2007) ENGLISH ET AL (2009) FENTON ET AL (2008) FERNELEY (1999) HEIJSTEK E CHAUDRON (2009) SCHNEIDEWIND E HINCHEY (2009)
Q13 – Periodicidade e Complexidade	Sem evidência adicional.
Q14 – Manutenibilidade e Confiabilidade	Sem evidência adicional.
Q15 – Periodicidade e Confiabilidade	Sem evidência adicional.

A Tabela 3-6 apresenta a quantidade total de evidências encontradas para cada questão, que representa a soma das evidências encontradas anteriormente por ARAÚJO (2009) e as evidências adicionais encontradas no contexto deste trabalho.

Tabela 3-6 – Quantidade de total de evidências para os relacionamentos apresentados no modelo de observação de evolução de software.

Questão – Características Relacionadas	# Evidências anteriores ARAÚJO (2009)	# Evidências adicionais	Total de Evidências por Questão
Q1 – Tamanho e Complexidade	3	2	**5**
Q2 – Tamanho e Confiabilidade	3	6	**9**
Q3 – Complexidade e Esforço	1	4	**5**
Q4 – Confiabilidade e Esforço	2	0	**2**
Q5 – Complexidade e Manutenibilidade	3	5	**8**
Q6 – Esforço e Manutenibilidade	1	1	**2**
Q7 – Periodicidade e Esforço	1	0	**1**
Q8 – Periodicidade e Manutenibilidade	0	0	**0**
Q9 – Tamanho e Esforço	5	7	**12**
Q10 – Tamanho e Manutenibilidade	0	4	**4**
Q11 – Periodicidade e Tamanho	1	0	**1**
Q12 – Complexidade e Confiabilidade	2	6	**8**
Q13 – Periodicidade e Complexidade	1	0	**1**
Q14 – Manutenibilidade e Confiabilidade	1	0	**1**
Q15 – Periodicidade e Confiabilidade	1	0	**1**
Total de Evidências	**25**	**35**	**60**

De acordo com os resultados obtidos através da reexecução do estudo, foi possível reafirmar 8 das 15 relações apresentadas no modelo proposto por ARAÚJO (2009), dentre elas o relacionamento entre as características; Q1 – Tamanho e Complexidade, Q2 – Tamanho e Confiabilidade, Q3 – Complexidade e Esforço, Q5 – Complexidade e Manutenibilidade, Q6 – Esforço e Manutenibilidade, Q8 – Periodicidade e Manutenibilidade (se manteve fora do modelo), Q9 – Tamanho e Esforço e Q12 – Complexidade e Confiabilidade.

É importante notar que mesmo dando uma maior abrangência na busca por evidências adicionais, através da inclusão da máquina de busca *Scopus* no protocolo da *quasi* revisão sistemática e também da atualização dos resultados da *EI Compendex*, não foram encontrados estudos na literatura técnica que abordassem o relacionamento entre as características Periodicidade e Manutenibilidade, referentes à questão de pesquisa Q8. Esse mesmo resultado foi encontrado na primeira execução do estudo (ARAÚJO 2009). A ausência de evidências para esse relacionamento nos permite reafirmar sua ausência no modelo original e, além disso, afirmar que até o momento não foi possível identificar nenhum estudo sobre ele na área de Engenharia de Software.

Além disso, os resultados mostram que existe um novo relacionamento a ser considerado pelo modelo, devido às evidências encontradas para a questão Q10, que envolve as características Tamanho e Manutenibilidade. De acordo com as 4 evidências adicionais encontradas para a questão de pesquisa Q10, CANFORA *et al.* (2005), CRUZ-LEMUS *et al.* (2007), KOZLOV *et al.* (2008) e RIAZ *et al.* (2009), quanto maior o tamanho, menor é a manutenibilidade.

Entre os 22 estudos selecionados como evidências para a atualização do modelo, apenas dois deles não apresentaram resultados que reafirmavam ou refutavam a existência das relações apresentadas no modelo, mas sugeriam como algumas delas deveriam ser exploradas.

O estudo apresentado por LEZACK (2005) afirma que utilizar somente a característica Tamanho (medida em linhas de código) para prever a Confiabilidade (medida em número de defeitos) não é uma boa estratégia para realizar a predição. Além disso, o estudo apresentado por DOLADO (2001) afirma que utilizar somente a característica Tamanho (medida em linhas de código) para prever o Esforço também não é uma boa estratégia para realizar a predição dessa característica.

Apesar desses dois estudos não reafirmarem e nem refutarem a existência do relacionamento entre as características Tamanho e Confiabilidade e Tamanho e Esforço, os resultados por eles apresentados favorecem o modelo original proposto por ARAÚJO (2009), tendo em vista que nele, a característica Confiabilidade é influenciada por outras quatro características e a característica Esforço é influenciada por outras três características.

Portanto, pode-se perceber que não foi encontrada nenhuma evidência que refutasse a existência de qualquer um dos relacionamentos analisados.

3.4 Evidências Adicionais para o Novo Relacionamento

Como mencionado anteriormente, as evidências adicionais encontradas na literatura técnica reafirmaram grande parte dos relacionamentos apresentados no modelo original (ARAÚJO 2009) e também sugeriram a adição de um novo relacionamento. Como descrito na Seção 3.3, todos os artigos utilizados como evidência para os relacionamentos apresentados no modelo passaram por um processo de análise detalhado com o objetivo de dar maior validade aos resultados apresentados.

Nessa seção serão apresentadas apenas as análises detalhadas realizadas nos artigos que possuíam evidências para o relacionamento entre Tamanho e Manutenibilidade (Q10), tento em vista que essas evidências podem causar um maior

impacto no modelo original e, por isso, devem ser discutidas antes de serem incluídas no modelo de observação de evolução de software.

Os artigos foram analisados do ponto de vista da questão de pesquisa Q10, ou seja, dando importância às características Tamanho e Manutenibilidade, embora alguns estudos apresentem evidências para outras questões.

No trabalho apresentado por CANFORA *et al.* (2005) foram realizados cinco estudos experimentais. Os grupos de participantes foram formados por estudantes, profissionais da indústria e da academia que possuíam experiência e conhecimento em modelagem de produtos de software, mas não tinham experiência ou conhecimentos sobre modelagem de processos de software (SPM). Uma sessão de treinamento foi realizada para fornecer aos participantes o conhecimento necessário para concluir as tarefas necessárias para o experimento. O objetivo do experimento era analisar estatisticamente quais métricas são úteis para indicar a manutenibilidade de modelos de processos de software. Foram utilizados 18 modelos de processos de software na execução do experimento. As métricas utilizadas para a característica Tamanho foram número o de atividades do modelo e número de artefatos do modelo. A métrica utilizada para medir a Manutenibilidade foi o tempo gasto na realização das atividades de manutenção do modelo. Para analisar os dados dos 18 modelos utilizados foi aplicado o teste de *Kolmogorov–Smirnov* com o objetivo de verificar se a distribuição dos dados coletados era normal. Como os dados não tinham uma distribuição normal, foi aplicada a análise de correlação de *Spearman,* com um nível de significância (α) de *5%,* que significa um nível de confiança de 95%. A principal diferença entre os cinco experimentos foi o perfil dos participantes, que variavam entre estudantes, profissionais da indústria e da academia. Os resultados dos cinco experimentos chegaram à conclusão de que existe um relacionamento entre Tamanho e Manutenibilidade dos modelos de processo de software, devido ao coeficiente de correlação ter sido maior do que 0.4684.

O trabalho apresentado por CRUZ-LEMUS *et al.* (2007) traz os resultados de um experimento que analisa a relação entre o Tamanho, medido através das métricas número de atividades e número de estados, e a Manutenibilidade, medida através da métrica de compreensibilidade (tempo, correção e completude), de diagramas de estado utilizando a notação UML. De acordo com a ISO/IEC 9126-3 (2003), a compreensibilidade é uma medida de qualidade interna da manutenibilidade. Para execução do experimento foram utilizados 20 diagramas de estados com diferentes valores para cada métrica analisada. O experimento foi executado com três diferentes grupos de participantes, onde o primeiro era composto por 10 professores e 8 estudantes de doutorado, o segundo grupo era formado por 24 estudantes e o terceiro

por 49 estudantes. Para analisar os dados foram calculadas as médias para cada uma das variáveis dependentes (tempo, correção e completude) para cada grupo de participantes. Para chegar aos resultados foi realizada uma análise de correlação de *Spearman* e os resultados obtidos mostram que existe uma alta relação entre Tamanho e Manutenibilidade.

No trabalho apresentado por KOZLOV *et al.* (2008), foi realizado um estudo no nível de código fonte com cinco sistemas de código aberto. O objetivo foi analisar a relação entre atributos internos de qualidade de software com a manutenibilidade de software. A métrica utilizada para Tamanho foi o número de linhas de código. As métricas utilizadas para Manutenibilidade foram o grau de modularidade, grau de portabilidade, grau de flexibilidade, grau de testabilidade, grau de legibilidade e grau de reusabilidade. Os valores para essas métricas foram calculados com base em métricas do código fonte, tais como, número de métodos, classes, atribuições, chamadas de função, entre outras. O relacionamento entre as características Tamanho e Manutenibilidade foi analisado através da análise de correlação de *Pearson.* O coeficiente de correlação de *Pearson* também foi utilizado para realizar uma análise linear, quadrática e cúbica do relacionamento entre as duas características. Os resultados do estudo apresentam a existência de um forte relacionamento entre o Tamanho do software e a sua Manutenibilidade. Os resultados mostram ainda que as curvas quadráticas e cúbicas não se encaixam nos pontos de forma significativamente melhor do que a curva linear, indicando que a curva linear é melhor para indicar o relacionamento analisado.

O trabalho apresentado por RIAZ *et al.* (2009) apresenta os resultados de uma revisão sistemática da literatura conduzida para coletar evidências sobre métricas e predição de manutenibilidade de software. Os resultados mostram que 5 dos 15 estudos analisados utilizam a métrica Tamanho para prever a Manutenibilidade do software.

3.5 Novo Modelo para Observação de Evolução de Software

Devido à boa qualidade dos estudos analisados com evidências adicionais para a questão de pesquisa Q10, que envolve as características de software Tamanho e Manutenibilidade, um novo relacionamento foi inserido no modelo original, resultando em um novo modelo de observação de evolução de software. A Figura 3-1 apresenta o Diagrama de Causas e Efeitos que descreve o novo modelo, atualizado com base aos resultados da evolução do estudo baseado em *quasi* revisão sistemática.

Na Figura 3-1, as setas tracejadas indicam os relacionamentos para os quais foram encontradas evidências adicionais que os reafirmaram, as setas sólidas indicam os relacionamentos para os quais não foram encontradas evidências adicionais e a seta pontilhada indica um novo relacionamento adicionado ao modelo devido às novas evidências encontradas.

Os relacionamentos para os quais não foram encontradas evidências adicionais foram mantidos no modelo devido às evidências apresentadas por ARAÚJO (2009) mostradas na Tabela 3-6.

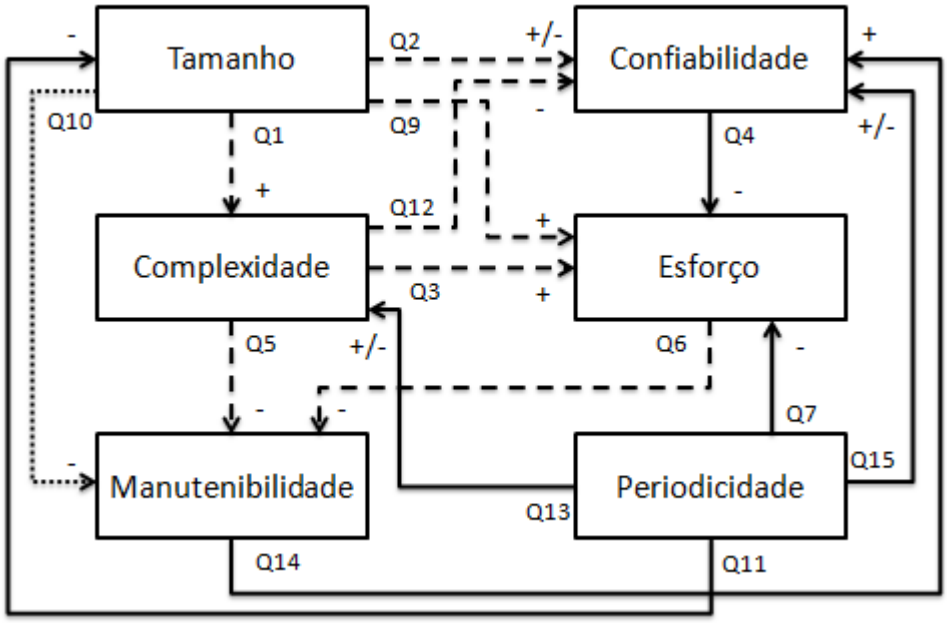


Figura 3-1 – Novo Modelo de Observação de Evolução de Software.

A partir da inclusão do novo relacionamento Q10 (Tamanho e Manutenibilidade) pode surgir a dúvida sobre sua equivalência com os relacionamentos Q1 (Tamanho e Complexidade), Q5 (Complexidade e Manutenibilidade). Entretanto, essa equivalência não pode ser confirmada com os estudos realizados no contexto deste trabalho, já que os relacionamentos entre as características foram analisados isoladamente para cada par de características.

A Tabela 3-7 apresenta um resumo dos resultados encontrados para cada questão com relação à alteração ou reafirmação do relacionamento analisado pela questão de pesquisa. A coluna “Status” possui os seguintes símbolos: “✓” caso o relacionamento tenha sido reafirmado através das evidências adicionais encontradas, “–” caso não tenham sido encontradas evidências adicionais e “I” caso o relacionamento tenha sido incluído devido às evidências adicionais encontradas.

Tabela 3-7 – Resultados para atualização dos relacionamentos entre as características.

Questão – Características Relacionadas	Status
Q1 – Tamanho e Complexidade	✓
Q2 – Tamanho e Confiabilidade	✓
Q3 – Complexidade e Esforço	✓
Q4 – Confiabilidade e Esforço	–
Q5 – Complexidade e Manutenibilidade	✓
Q6 – Esforço e Manutenibilidade	✓
Q7 – Periodicidade e Esforço	–
Q8 – Periodicidade e Manutenibilidade	✓
Q9 – Tamanho e Esforço	✓
Q10 – Tamanho e Manutenibilidade	I
Q11 – Periodicidade e Tamanho	–
Q12 – Complexidade e Confiabilidade	✓
Q13 – Periodicidade e Complexidade	–
Q14 – Manutenibilidade e Confiabilidade	–
Q15 – Periodicidade e Confiabilidade	–

Tendo em vista que o objetivo do estudo de *quasi* revisão sistemática foi de atualizar o corpo de conhecimento sobre os relacionamentos entre as características de software utilizadas na construção do modelo analisado e verificar se as evidências apresentadas por ARAÚJO (2009) mantêm sua validade, pode-se concluir que os resultados encontrados foram importantes para a atualização do modelo. A partir desses resultados foi possível realizar novos estudos experimentais de simulação para comparar as alterações propostas e avaliar os resultados dessas alterações.

3.6 Conclusão

O objetivo principal deste capítulo foi apresentar os resultados da *quasi* revisão sistemática realizada para encontrar evidências adicionais sobre os relacionamentos apresentados no modelo de observação de evolução de software.

Ao todo foram selecionados 22 artigos que apresentavam 35 evidências adicionais. Dentre os 13 relacionamentos apresentados no modelo, 8 deles foram reafirmados, um novo relacionamento, entre as características Tamanho e Manutenibilidade, foi incluído e nenhum dos relacionamentos apresentados no modelo original foi refutado.

No próximo capítulo serão apresentados os resultados de simulações executadas com o novo modelo de observação de evolução de software. Além disso, serão apresentadas comparações entre resultados obtidos com o modelo original e com o novo modelo.

4 Estudos Experimentais em Evolução de Software

Este capítulo apresenta as alterações realizadas no modelo original de Dinâmica de Sistemas proposto por ARAÚJO (2009), os resultados da simulação realizada com o novo modelo, uma comparação entre os resultados obtidos da simulação com o modelo original e com o novo modelo, os resultados obtidos com a simulação do novo modelo utilizando novos dados do projeto previamente analisado, uma análise do impacto das alterações realizadas no modelo e também uma comparação entre os resultados apresentados pelas ferramentas *Illium Software Evolution* e Vensim PLE.

4.1 Novo Modelo de Dinâmica de Sistemas para Observação de Evolução de Software

As alterações realizadas no modelo original de Dinâmica de Sistemas foram divididas em duas etapas. Na primeira, as alterações estão relacionadas à atualização conceitual do modelo apresentado no Capítulo 3. Na segunda, as alterações estão relacionadas a uma análise crítica realizada sobre o modelo de Dinâmica de Sistemas, com objetivo de ampliar as possibilidades de observar novos comportamentos que serão discutidos posteriormente.

A primeira modificação realizada no modelo original de Dinâmica de Sistemas (ARAÚJO 2009) foi a inclusão do novo relacionamento sugerido pelas evidências adicionais encontradas pelo estudo secundário de *quasi* revisão sistemática, executado no contexto deste trabalho. O novo relacionamento envolve as características de software Tamanho e Manutenibilidade.

Durante a inclusão do novo relacionamento foi identificado que a modelagem de Dinâmica de Sistemas para o modelo original, apresentada por ARAÚJO (2009), não considerava o relacionamento entre as características Esforço e Manutenibilidade, que apesar de previsto no modelo conceitual original, não foi considerada na simulação.

Observando o modelo conceitual original (ARAÚJO 2009), percebe-se que o relacionamento entre as características Esforço e Manutenibilidade permite a formação de um *loop* de realimentação (*feedback*) entre as características Esforço, Manutenibilidade e Confiabilidade. De acordo com MADACHY (2008), um *loop* se refere a uma cadeia fechada de relacionamento. Um *loop* de realimentação também pode ser identificado como uma referência circular entre variáveis. A Figura 4-1

apresenta o novo modelo conceitual de observação de evolução de software destacando o *loop* de realimentação contido no modelo através de linhas escuras.

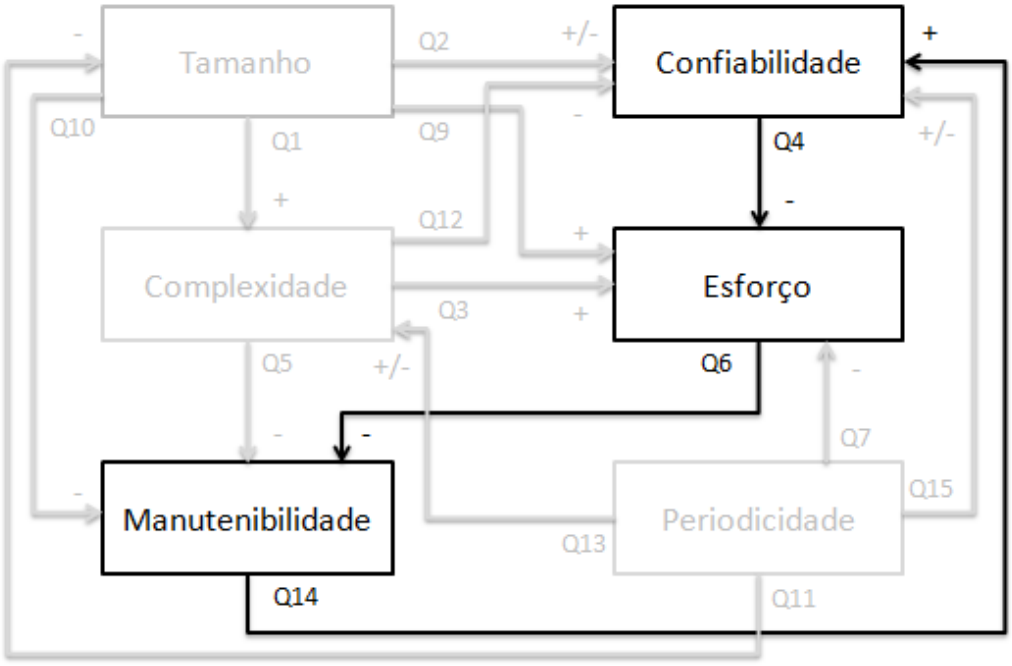


Figura 4-1 – Novo Modelo de Observação de Evolução de Software destacando o *loop* de realimentação.

A ausência do relacionamento entre as características Esforço e Manutenibilidade, no modelo de Dinâmica de Sistemas apresentado por ARAÚJO (2009) pode ser explicada pela dificuldade de modelar *loops* entre variáveis auxiliares utilizando a ferramenta *Illium Software Evolution*, devido à limitação das funcionalidades disponíveis na ferramenta que não possibilitam diretamente a modelagem de *loops*.

4.1.1 Modelagem de *Loops* em Dinâmica de Sistemas

No modelo de Dinâmica de Sistemas construído por ARAÚJO (2009), o relacionamento entre as características de software analisadas é representado através de variáveis auxiliares. Através dessas variáveis é realizado o cálculo da influência entre as características. Entretanto, não é possível incluir o relacionamento entre Esforço e Manutenibilidade através de variáveis auxiliares utilizando a ferramenta *Illium Software Evolution*. Para representar um *loop* em Dinâmica de Sistemas, na linguagem utilizada por *Illium*, é necessário utilizar variáveis do tipo *Stock (Level)*, caso contrário, a ferramenta de simulação acusa um erro de referência circular no modelo e não permite que a execução do modelo seja realizada. Lembrando que uma variável do tipo *Stock* é uma variável responsável por acumular valores e que permite definir

um valor inicial. Em *Illium*, não é possível definir um valor inicial para as variáveis auxiliares, o que também dificulta a implementação de um *loop* na ferramenta.

Para incluir o relacionamento entre as características Esforço e Manutenibilidade, que forma um *loop* entre algumas variáveis, foi necessário alterar a modelagem de Dinâmica de Sistemas. Para isso, foi realizada uma pesquisa na literatura técnica sobre modelagem em Dinâmica de Sistemas e foram encontradas algumas soluções para resolver o caso da referência circular.

A primeira solução encontrada, apresentada por KIRKWOOD (1998), foi a utilização da função SMOOTHI, que é fornecida pela ferramenta Vensim® PLE. Essa função tem o mesmo objetivo da função SMOOTH, que, de acordo com VENSIM (2003a), é usada para representar expectativas, entretanto se difere apenas pelo fato de aceitar um valor inicial, o que possibilita a execução de modelos que implementam *loops* através de variáveis auxiliares.

A função SMOOTHI recebe como parâmetros um valor de entrada, um valor de *delay* e um valor inicial. No primeiro passo da simulação a função utiliza o valor inicial para calcular as equações envolvidas no *loop* e assim consegue executar o modelo (VENSIM 2003a). Os resultados obtidos com essa solução foram coerentes com os resultados obtidos com a modelagem que utilizou somente variáveis auxiliares e uma variável do tipo *Stock*, descrita posteriormente na terceira solução, já que a única alteração ocorreu nos valores da variável passada para a função, neste caso a variável auxiliar utilizada para calcular a variação do Esforço. A alteração nos valores ocorreu devido ao *delay* utilizado pela função, gerando uma diferença insignificante para o resultado final, tendo em vista que a tendência das características é o foco do estudo, e não os valores.

A segunda solução encontrada, apresentada por (VENSIM 2003b), foi a utilização da função SIMULTANEOUS, também fornecida pela ferramenta Vensim® PLE. Esta função recebe dois parâmetros, onde o primeiro é a expressão formada pelas variáveis que envolvem *loop*, e o segundo, é uma expressão de inicialização, que geralmente é uma constante representando o primeiro parâmetro. No momento inicial da simulação, a expressão passada no primeiro parâmetro para a função será inicializada com o valor passado como segundo parâmetro, que será usado para calcular todas as equações envolvidas no *loop* até que os valores das variáveis dessas equações não mudem significativamente (VENSIM 2003b). Uma dificuldade encontrada para utilizar essa funcionalidade é o cálculo do segundo parâmetro, já que as expressões das variáveis que compõem o *loop* têm pelo menos três termos com coeficientes calculados através de regressão linear, o que dificulta a aplicação dessa solução caso a modelagem seja feita manualmente.

Como este trabalho tem o objetivo de utilizar a máquina de simulação contida em *Illium* (BARROS *et al.* 2000), a mesma utilizada por ARAÚJO (2009), foi necessário implementar as soluções encontradas na literatura utilizando essa ferramenta. Entretanto, foi identificado que *Illium* não implementa as duas funções, encontradas na literatura técnica como solução para o caso da referência circular (*loops*). Portanto, outras soluções foram investigadas para solucionar o problema.

A princípio, a possibilidade de implementar uma das duas funcionalidades, descritas como solução para a questão dos loops, na máquina de simulação de *Illium* (BARROS *et al.* 2000) foi analisada em conjunto com o autor da ferramenta. Entretanto, chegou-se a conclusão que não bastaria apenas implementar uma das soluções na ferramenta. Além disso, seria necessário alterar o analisador sintático da ferramenta, que identifica a dependência entre as variáveis de forma estática. Portanto, a solução seria alterar a modelagem de Dinâmica de Sistemas.

De acordo com o manual da ferramenta Vensim (VENSIM 2003a), a implementação da função SMOOTHI nada mais é do que a utilização de uma variável do tipo *Stock (Level)* auxiliando o cálculo da variável. A equação da funcionalidade é apresentada na Equação 2,

SMOOTHI= ((*input* - SMOOTHI)/*delay*, *initial value*) (Eq. 2)

onde SMOOTHI é uma variável do tipo STOCK, "*input*" representa o valor das taxas (*Rates*) de entrada, "*delay*" o tempo de atraso para calcular o valor e "initial value" é o valor inicial definido para a variável SMOOTHI.

Dessa forma, a partir dos detalhes de implementação da funcionalidade SMOOTHI, disponíveis na literatura técnica, foi possível modificar a modelagem para obter o mesmo comportamento obtido pela utilização dessa funcionalidade.

Portanto, a terceira solução implementada para resolver a questão do *loop*, sem utilizar as funcionalidades sugeridas na literatura técnica, foi alterar a modelagem de modo que pelo menos uma das variáveis que compõem o *loop* deixe de ser uma variável auxiliar e passe a ser tratada como uma variável do tipo *Stock (Level),* e que a cada passo de simulação essa variável seja zerada, ou seja, nenhum valor seja acumulado. Transformando uma das variáveis auxiliares que formam o *loop* em uma variável do tipo *Stock (Level)* é possível atribuir um valor inicial a ela para que seja utilizado no primeiro passo da simulação e assim resolver a questão da referência circular.

A Figura 4-2 representa a modelagem em Dinâmica de Sistemas, utilizando a ferramenta Vensim PLE, do novo modelo de observação de evolução de software considerando todos os relacionamentos apresentados.

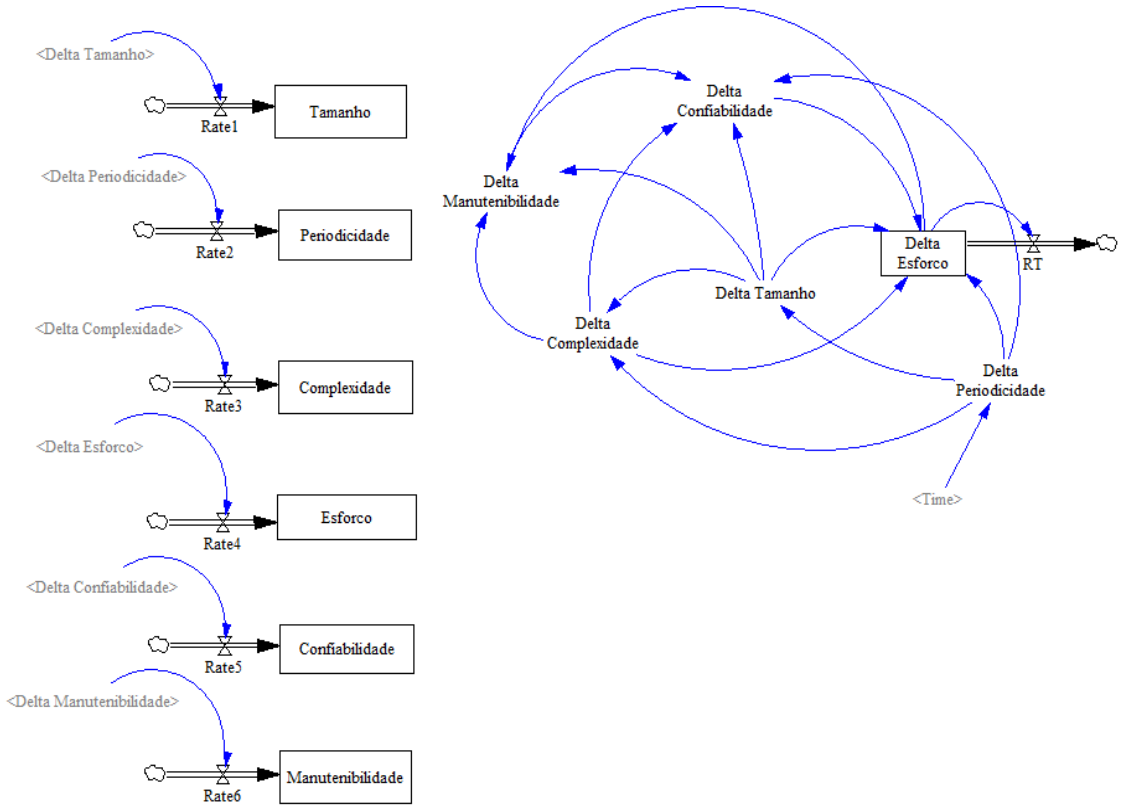


Figura 4-2 – Modelo de Dinâmica de Sistemas representando o Modelo de Observação de Evolução de Software.

Na Figura 4-2, as *Rates* de 1 a 6 representam a taxa de crescimento de cada *Stock (Level)*. A *Rate* RT representa a equação que reinicia a variável a cada passo da simulação. Observa-se que a variável "Delta Esforço" é a única variável do tipo *Stock* a fazer parte do cálculo da influência entre as variáveis. Essa solução teve que ser implementada para ser utilizada para representar *loops* em *Illium*, ressaltando que na ferramenta Vensim PLE outras soluções são possíveis.

Uma comparação entre o modelo original e o novo modelo de Dinâmica de Sistemas é apresentada na Tabela 4-1 utilizando a linguagem definida em *Illium* (BARROS *et al.* 2000). As alterações realizadas na modelagem encontram-se sublinhadas na coluna da direita. As equações apresentadas foram calculadas com os dados utilizados por ARAÚJO (2009).

Tabela 4-1 – Primeira etapa de modificações. Comparação entre a Modelagem Original e a Nova Modelagem.

Modelagem Original (ARAÚJO 2009)	Nova Modelagem – Primeiras Modificações
SPEC DT 1.00; ################################## # SOFTWARE CHARACTERISTICS ################################## **STOCK** SIZE 63.23; **STOCK** PERIODICITY 4.00; **STOCK** COMPLEXITY 11.43; **STOCK** EFFORT 0.50; **STOCK** RELIABILITY 1.00; **STOCK** MAINTAINABILITY 0.50; **STOCK** TIMER 1.00; ############################## # RATES ############################## **PROC** DELTA_PERIODICITY 0.23 * TIMER; **PROC** DELTA_SIZE 0.03 * DELTA_PERIODICITY; **PROC** DELTA_COMPLEXITY 0.07 * DELTA_SIZE + 0.01 * DELTA_PERIODICITY; **PROC** DELTA_MAINTAINABILITY -11.95 * DELTA_COMPLEXITY; **PROC** DELTA_RELIABILITY -0.92 * DELTA_SIZE + 0.20 * DELTA_PERIODICITY + 2.24 * DELTA_COMPLEXITY + 0.17 * DELTA_MAINTAINABILITY; **PROC** DELTA_EFFORT -8.18 * DELTA_SIZE + 0.39 * DELTA_PERIODICITY + (-75.36 * DELTA_COMPLEXITY) + 4.55 * DELTA_RELIABILITY; **RATE** (**SOURCE**, **TIMER**) RTT DT; **RATE** (**SOURCE**, PERIODICITY) RTP **MAX**(0,DELTA_PERIODICITY) ; **RATE** (**SOURCE**, SIZE) RTS **MAX** (0, DELTA_SIZE) ; **RATE** (**SOURCE**, COMPLEXITY) RTC **MAX** (0, DELTA_COMPLEXITY) ; **RATE** (**SOURCE**, MAINTAINABILITY) RTM **MAX** (0, DELTA_MAINTAINABILITY) ; **RATE** (**SOURCE**, RELIABILITY) RTR **MAX** (0, DELTA_RELIABILITY) ; **RATE** (**SOURCE**, EFFORT) RTE **MAX** (0, DELTA_EFFORT);	**SPEC** DT 1.00; ################################## # SOFTWARE CHARACTERISTICS ################################## **STOCK** SIZE 63.23; **STOCK** PERIODICITY 4.00; **STOCK** COMPLEXITY 11.43; **STOCK** EFFORT 0.50; **STOCK** RELIABILITY 1.00; **STOCK** MAINTAINABILITY 0.50; **STOCK** TIMER 1.00; **STOCK** DELTA_EFFORT 1.00; ################################## # RATES ################################## **PROC** DELTA_PERIODICITY 0.23 * TIMER; **PROC** DELTA_SIZE 0.03 * DELTA_PERIODICITY; **PROC** DELTA_COMPLEXITY 0.07 * DELTA_SIZE + 0.01 * DELTA_PERIODICITY; **PROC** DELTA_MAINTAINABILITY -11.95 * DELTA_COMPLEXITY + 1.01 * DELTA_EFFORT + (-4.38 * DELTA_SIZE); **PROC** DELTA_RELIABILITY -0.92 * DELTA_SIZE + 0.20 * DELTA_PERIODICITY + 2.24 * DELTA_COMPLEXITY + 0.17 * DELTA_MAINTAINABILITY; **PROC** DELTA_EFFORT_AUX -8.18 * DELTA_SIZE + 0.39 * DELTA_PERIODICITY + (-75.36 * DELTA_COMPLEXITY) + 4.55 * DELTA_RELIABILITY; **RATE** (**SOURCE**, **TIMER**) RTT DT; **RATE** (**SOURCE**, PERIODICITY) RTP **MAX**(0,DELTA_PERIODICITY) ; **RATE** (**SOURCE**, SIZE) RTS **MAX** (0, DELTA_SIZE) ; **RATE** (**SOURCE**, COMPLEXITY) RTC **MAX** (0, DELTA_COMPLEXITY) ; **RATE** (**SOURCE**, MAINTAINABILITY) RTM **MAX** (0,DELTA_MAINTAINABILITY) ; **RATE** (SOURCE, RELIABILITY) RTR MAX (0, DELTA_RELIABILITY) ; **RATE** (SOURCE, EFFORT) RTE MAX (0, DELTA_EFFORT); **RATE** (**SOURCE**, DELTA_EFFORT) RTDEO DELTA_EFFORT_AUX - DELTA_EFFORT;

Portanto, a primeira etapa de modificações na modelagem de Dinâmica de Sistema consistiu em:

- Incluir o termo X * Delta_Esforço + Y * Delta_Tamanho na expressão que calcula a influência sobre variável Delta_Manutenibilidade, onde X e Y são os coeficientes calculados a partir de regressão linear. Essas modificações são referentes à inclusão do novo relacionamento e o relacionamento que estava ausente da modelagem;
- Incluir uma variável do tipo *Stock* para auxiliar o cálculo da variável esforço com o objetivo de resolver a questão do *loop.*

A modelagem de dinâmica de sistemas apresentada considera cada uma das características Tamanho, Periodicidade, Complexidade, Esforço e Manutenibilidade apresentadas no novo modelo, como variáveis do tipo *Stock (Level)*, que são variáveis que acumulam valores.

Na linguagem de modelagem de dinâmica de sistemas definida em *Illium* as variáveis do tipo *Stock* são definidas da seguinte maneira:

STOCK SIZE <valor inicial>;

O valor definido após o nome da variável, neste caso chamada de SIZE, representa o valor inicial do *Stock,* que é obtido pelo valor da última versão do sistema utilizada na simulação para a característica em questão. O valor da última versão é considerado como valor inicial de cada *Stock* para definir que o ponto de partida da simulação são os valores da última versão analisada do sistema.

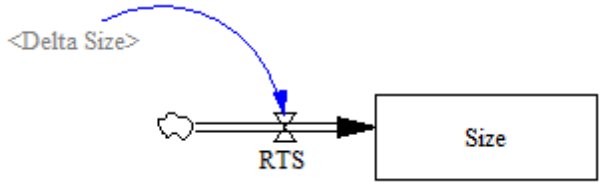
As taxas (*Rate*) que definem como as variáveis do tipo *Stock* acumulam os valores são calculadas através de variáveis auxiliares chamadas de *delta_tamanho*, *delta_esforço*, e assim respectivamente para cada característica presente no modelo. Em *Illium* as taxas são definidas da seguinte maneira:

RATE (**SOURCE**, SIZE) RTS **MAX** (0, DELTA_SIZE);

Uma variável do tipo *Rate* contêm dois parâmetros, onde o primeiro representa a fonte e o segundo representa o destino daquela taxa. A palavra reservada *Source* representa uma fonte infinita externa ao modelo. A palavra reservada MAX representa uma função matemática definida por *Illium*, onde o maior valor entre os dois parâmetros passados para a função é retornado. Nesse caso, RTS representa o nome da *Rate*. Para facilitar a compreensão e fazer uma comparação entre modelagens, a

Tabela 4-2 apresenta uma correspondência entre a modelagem definida em *Illium* e a modelagem gráfica definida pela ferramenta Vensim PLE.

Tabela 4-2 – Correspondência entre a modelagem em *Illium* e Vensim PLE.

Illium	**Vensim PLE**
RATE (**SOURCE**, SIZE) RTS **MAX** (0, DELTA_SIZE)	

Na ferramenta Vensim PLE a definição da equação de uma *Rate* não é apresentada na tela de modelagem. Para definir uma equação em Vensim PLE uma nova janela é apresentada, como mostrado na Figura 4-3.

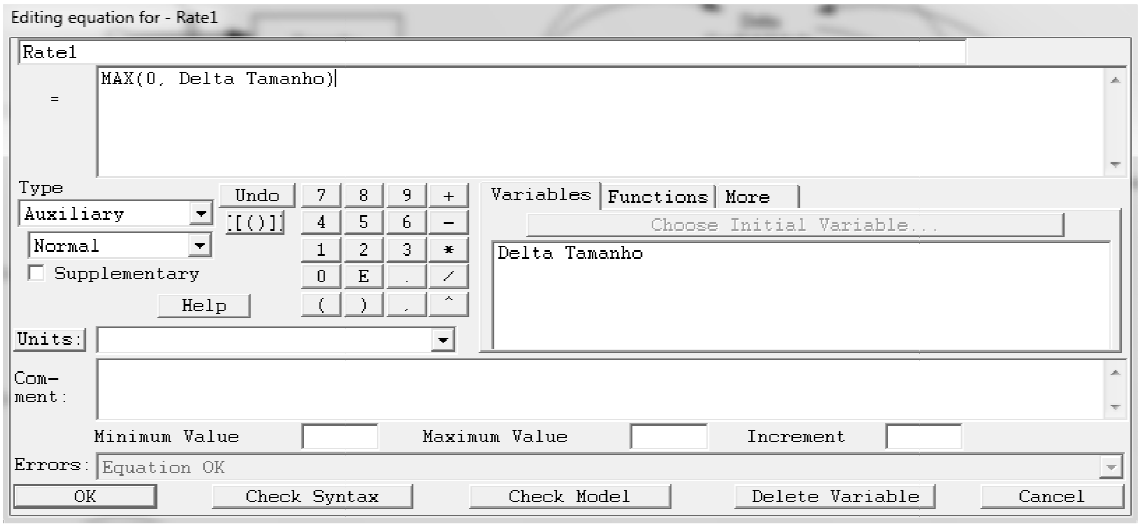


Figura 4-3 – Janela para edição da equação de uma Rate usando Vensim (VENSIM 2011).

As influências entre as características de software, representadas pelas setas que interligam as características na Figura 4-2 foram modeladas utilizando variáveis auxiliares. Essas variáveis são definidas em *Illium* como procedimentos, que utilizam a palavra reservada **PROC**. Por exemplo, em *Illium* o valor da variável auxiliar DELTA_SIZE é calculado da seguinte maneira:

PROC DELTA_SIZE 0.03 * DELTA_PERIODICITY;

Onde a variável DELTA_PERIODICITY representa o relacionamento entre a característica Tamanho e Periodicidade apresentado no modelo e o coeficiente desta equação é calculado através de regressão linear aplicada sobre os dados de Tamanho e Periodicidade das versões analisadas do projeto observado neste trabalho.

Em Vensim PLE a equação correspondente ao Stock SIZE é apresentada na Figura 4-4.

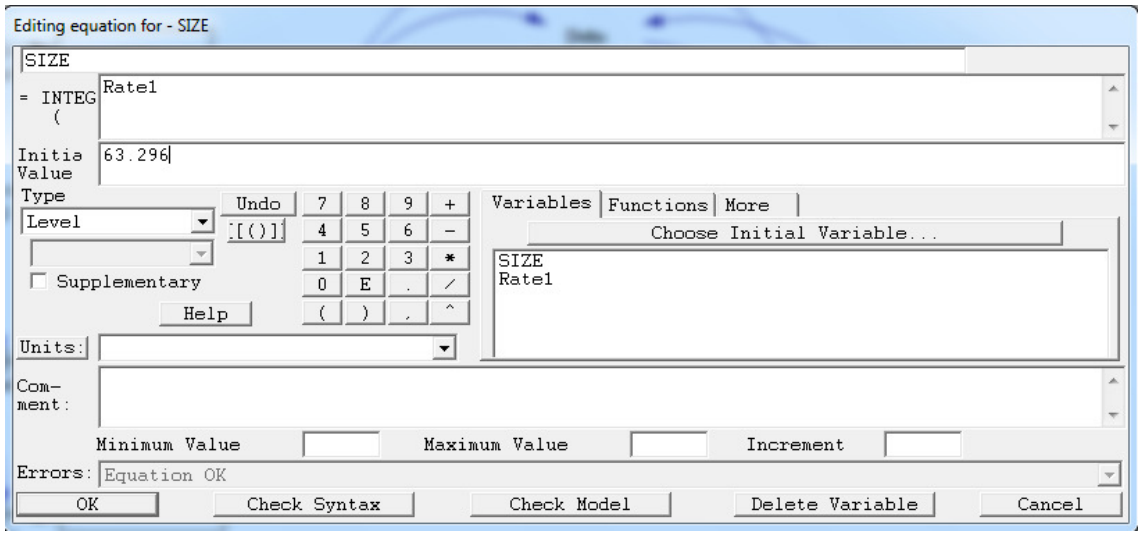


Figura 4-4 – Janela para edição da equação de uma variável do tipo Stock em Vensim PLE (VENSIM 2011).

A cada passo de simulação, o valor de cada variável DELTA, definida para cada uma das características de software analisadas, é associado ao fluxo de entrada de uma variável *Stock* correspondente. Por exemplo, o valor da variável auxiliar DELTA_SIZE é utilizado para calcular o fluxo de entrada da variável Size, que é do tipo *Stock.*

4.1.2 Análise Crítica sobre o Modelo de Dinâmica de Sistemas

Após as alterações realizadas no modelo de Dinâmica de Sistemas original para considerar o relacionamento entre as características Esforço e Manutenibilidade e incluir o novo relacionamento entre as características Tamanho e Manutenibilidade na modelagem, foi realizada uma análise crítica sobre a modelagem de Dinâmica de Sistemas que levou a uma segunda etapa de modificações.

O modelo de observação da evolução de software apresentado na Figura 3-1, através do Diagrama de Causas e Efeitos, mostra como cada característica de software analisada influencia a outra de acordo com evidências encontradas na literatura. A representação deste modelo através de Dinâmica de Sistemas resultou em um modelo que calcula a tendência de crescimento para cada característica através dos relacionamentos apresentados no modelo de observação de evolução de software.

Analisando como é realizado o cálculo do fluxo de entrada (*Rate*) de cada característica de software (variáveis *Stock*), pode-se perceber que os valores de cada característica jamais poderão diminuir independente de qual seja a tendência dos dados reais de uma característica apresentada por um sistema. Por exemplo, em um determinado projeto analisado foi observado que o tamanho do sistema diminui a cada iteração e, portanto, aplicando a regressão linear nos dados, através do método dos mínimos quadrados, obtêm-se um coeficiente negativo que representará a tendência de diminuição do tamanho. Entretanto, esta tendência jamais poderá ser observada através do modelo de dinâmica de sistemas apresentado em ARAÚJO (2009), já que a taxa de alteração de cada *Rate* utiliza a função matemática MAX com o primeiro parâmetro igual a zero e no segundo parâmetro o valor calculado através das influências entre as características, que pode ser negativo caso a tendência seja de diminuição. Ou seja, o valor que será utilizado para alterar um *Stock* nunca será negativo. De acordo com ARAÚJO (2009), a utilização desta função teve o objetivo de evitar que um *Stock* adquirisse valores negativos, já que não faz sentido que nenhuma das características possua valores negativos, como por exemplo, um Tamanho negativo. Entretanto, essa solução impossibilita a observação de certos comportamentos da evolução de um sistema.

Do ponto de vista prático, realmente não faz sentido o valor de nenhuma característica apresentada pelo modelo ser negativo. Por outro lado, faz sentido a tendência de crescimento de uma das características ser negativa, ou seja, diminuir com o tempo. Entretanto, a modelagem apresentada, além de impedir que os valores das características sejam negativos, também impossibilita que a tendência de qualquer uma das características seja de diminuição.

Para resolver esse problema foi necessário fazer algumas alterações na lógica do cálculo dos fluxos de entrada de cada característica para ampliar as possibilidades de observação de outros comportamentos da evolução de um sistema, que não podiam ser observados com a modelagem original proposta por ARAÚJO (2009).

A segunda etapa de alterações consistiu em remover a utilização da funcionalidade MAX nas equações das taxas das *Rates* de cada característica e incluir a seguinte condição em cada *Rate.*

Se ((VARIÁVEL_STOCK <= 0) AND (DELTA_NOME_DA_VARIÁVEL < 0)) então *Rate* recebe (-VARIÁVEL_STOCK) senão recebe DELTA_NOME_DA_VARIÁVEL.

Onde VARIÁVEL_STOCK é o nome da variável *Stock* para cada característica analisada e DELTA_NOME_DA_VARIÁVEL é a variável auxiliar utilizada para calcular

a taxa de cada *Rate* de acordo com as variáveis que influenciam determinada característica.

Dessa forma, caso o *Stock* de uma característica tenha atingido um valor negativo e a tendência continua a ser de diminuição, então o valor do *Stock* será alterado para o valor zero, impedindo que o valor final seja negativo. E caso o *Stock* não seja negativo, mas a taxa de uma *Rate* seja negativa (indicando uma diminuição) o valor do *Stock* irá diminuir de acordo com o valor da *Rate*, permitindo assim que os valores das características diminuam ao longo do tempo. Portanto, dessa forma, será possível observar uma tendência de diminuição das características de software, caso os dados históricos utilizados apresentem esse comportamento e os valores finais das características jamais serão negativos.

Além disso, outra modificação realizada no modelo foi a inclusão do termo de multiplicação de cada RATE definida no modelo pela constante DT. Dessa forma, o usuário pode definir a variação do tempo em cada passo de simulação.

A Tabela 4-3 mostra duas versões do modelo de Dinâmica de Sistemas para facilitar a visualização das modificações realizadas ao longo do trabalho. A primeira coluna apresenta a versão do modelo resultante da primeira etapa de modificações, apresentada anteriormente na Tabela 4-1, e a segunda coluna apresenta a versão resultante da segunda etapa de modificações na modelagem. As alterações realizadas na modelagem estão sublinhadas na segunda coluna.

Tabela 4-3 – Segunda etapa de modificações. Comparação entre o Modelo de Dinâmica de Sistemas, Modelagem com a primeira e a segunda etapa de modificações.

Nova Modelagem – Primeira etapa de modificações	**Nova Modelagem – Segunda etapa de modificações.**
##################################	##################################
SPEC DT 1.00;	**SPEC** DT 1.00;
##################################	##################################
# SOFTWARE CHARACTERISTICS	# SOFTWARE CHARACTERISTICS
##################################	##################################
STOCK SIZE 63.23;	**STOCK** SIZE 63.23;
STOCK PERIODICITY 4.00;	**STOCK** PERIODICITY 4.00;
STOCK COMPLEXITY 11.43;	**STOCK** COMPLEXITY 11.43;
STOCK EFFORT 0.50;	**STOCK** EFFORT 0.50;
STOCK RELIABILITY 1.00;	**STOCK** RELIABILITY 1.00;
STOCK MAINTAINABILITY 0.50;	**STOCK** MAINTAINABILITY 0.50;
STOCK TIMER 1.00;	**STOCK** TIMER 1.00;
STOCK DELTA_EFFORT 0.50;	**STOCK** DELTA_EFFORT 0.50;
##################################	##################################
# RATES	# RATES
##################################	##################################
PROC DELTA_PERIODICITY 0.23 * TIMER;	**PROC** DELTA_PERIODICITY 0.23 * TIMER;
PROC DELTA_SIZE 0.03 * DELTA_PERIODICITY;	**PROC** DELTA_SIZE 0.03 * DELTA_PERIODICITY;
PROC DELTA_COMPLEXITY 0.07 * DELTA_SIZE + 0.01 * DELTA_PERIODICITY;	**PROC** DELTA_COMPLEXITY 0.07 * DELTA_SIZE + 0.01 * DELTA_PERIODICITY;

PROC DELTA_MAINTAINABILITY -11.95 * DELTA_COMPLEXITY + 1.01 * DELTA_EFFORT + (-4.38 * DELTA_SIZE);	**PROC** DELTA_MAINTAINABILITY -11.95 * DELTA_COMPLEXITY + 1.01 * DELTA_EFFORT + (-4.38 * DELTA_SIZE);
PROC DELTA_RELIABILITY -0.92 * DELTA_SIZE + 0.20 * DELTA_PERIODICITY + 2.24 * DELTA_COMPLEXITY + 0.17 * DELTA_MAINTAINABILITY;	**PROC** DELTA_RELIABILITY -0.92 * DELTA_SIZE + 0.20 * DELTA_PERIODICITY + 2.24 * DELTA_COMPLEXITY + 0.17 * DELTA_MAINTAINABILITY;
PROC DELTA_EFFORT_AUX -8.18 * DELTA_SIZE + 0.39 * DELTA_PERIODICITY +(-75.36 * DELTA_COMPLEXITY) + 4.55 * DELTA_RELIABILITY;	**PROC** DELTA_EFFORT_AUX -8.18 * DELTA_SIZE + 0.39 * DELTA_PERIODICITY + (-75.36 * DELTA_COMPLEXITY) + 4.55 * DELTA_RELIABILITY;
RATE (**SOURCE**, **TIMER**) RTT DT;	**RATE** (**SOURCE**, TIMER) RTT DT;
RATE (**SOURCE**, PERIODICITY) RTP **MAX**(0,DELTA_PERIODICITY) ;	**RATE** (**SOURCE**, PERIODICITY) RTP **IF** (**AND** (PERIODICITY <= 0, DELTA_PERIODICITY <0), -PERIODICITY, DELTA_PERIODICITY) *** DT**;
RATE (**SOURCE**, SIZE) RTS **MAX** (0, DELTA_SIZE) ;	**RATE** (**SOURCE**, SIZE) RTS **IF** (**AND** (SIZE <= 0, DELTA_SIZE <0), -SIZE , DELTA_SIZE) *** DT**;
RATE (**SOURCE**, COMPLEXITY) RTC **MAX** (0, DELTA_COMPLEXITY) ;	**RATE** (**SOURCE**, COMPLEXITY) RTC **IF** (**AND** (COMPLEXITY <= 0, DELTA_COMPLEXITY <0), -COMPLEXITY, DELTA_COMPLEXITY) *** DT**;
RATE (**SOURCE**, MAINTAINABILITY) RTM **MAX** (0,DELTA_MAINTAINABILITY) ;	**RATE**(SOURCE, MAINTAINABILITY) RTM **IF** (**AND** (MAINTAINABILITY <= 0, DELTA_MAINTAINABILITY <0), -MAINTAINABILITY, DELTA_MAINTAINABILITY) *** DT**;
RATE (SOURCE, RELIABILITY) RTR MAX (0, DELTA_RELIABILITY) ;	**RATE** (**SOURCE**, RELIABILITY) RTR **IF** (**AND** (RELIABILITY <= 0, DELTA_RELIABILITY <0), -RELIABILITY , DELTA_RELIABILITY) *** DT**;
RATE (SOURCE, EFFORT) RTE MAX (0, DELTA_EFFORT);	**RATE** (**SOURCE**, EFFORT) RTE **IF** (**AND** (EFFORT <= 0, DELTA_EFFORT <0), -EFFORT , DELTA_EFFORT) *** DT**;
RATE (**SOURCE**, DELTA_EFFORT) RTDEO DELTA_EFFORT_AUX - DELTA_EFFORT;	**RATE** (**SOURCE**, DELTA_EFFORT) RTDEO (DELTA_EFFORT_AUX - DELTA_EFFORT);

Portanto, a segunda etapa de modificações na modelagem de Dinâmica de Sistema consistiu em:

- Alterar a lógica do cálculo das *Rates* de cada característica de software analisada, deixando de usar a função MAX e passando a usar a **IF** e **AND**, e também, multiplicar cada Rate por **DT** . Por exemplo, para a taxa (*Rate*) de entrada da variável SIZE: **IF** (**AND** (SIZE <= 0, DELTA_SIZE <0), -SIZE, DELTA_SIZE) *** DT**;

É importante ressaltar que as mudanças realizadas na segunda etapa não apresentam alterações nos resultados das simulações realizadas com o conjunto de dados utilizados por ARAÚJO (2009). O objetivo dessas alterações é retratar o

fenômeno da evolução de software de forma mais fidedigna através do modelo de Dinâmica de Sistemas. Por isso, as alterações foram feitas de forma a possibilitar a observação de novos comportamentos, dependendo do conjunto de dados utilizado para construir as equações do modelo.

4.2 Simulação com o Novo Modelo (Primeira Coleta de Dados)

Após realizar as modificações no modelo conceitual de observação de evolução de software, proposto por ARAÚJO (2009), através das evidências adicionais encontradas na literatura técnica de Engenharia de Software e refletir essas alterações no modelo de Dinâmica de Sistemas, foram executados estudos experimentais de simulação com o novo modelo utilizando um conjunto de dados apresentados por ARAÚJO (2009). Antes de apresentar a simulação com o novo modelo, será apresentada uma análise dos dados utilizados para calcular as equações do modelo de simulação. A primeira coleta de dados representa aquela realizada por ARAÚJO (2009).

4.2.1 Análise dos Dados Utilizados na Simulação

Para comparar os resultados da simulação obtidos com o modelo original e com o novo modelo, decidiu-se por utilizar os mesmos dados do projeto utilizado por ARAÚJO (2009) para executar as simulações.

O projeto analisado corresponde a uma aplicação Web de grande porte que controla o fluxo de trabalho para a execução de projetos, tanto em nível financeiro quanto administrativo e operacional. O desenvolvimento do sistema conta com equipes distribuídas geograficamente, utilizando a linguagem Java para o desenvolvimento da aplicação. O processo de desenvolvimento é iterativo e incremental, baseado nos níveis de G a C do modelo de referência MPS.BR (2009). A qualidade do sistema é uma preocupação contínua, sendo submetido a avaliações de qualidade através de técnicas de verificação, validação e teste. A equipe é estável, contando com cerca de 12 desenvolvedores (ARAÚJO 2009).

De acordo com (ARAÚJO 2009), inicialmente foram consideradas 16 versões do sistema, disponíveis nos repositórios de controle de versão do sistema. Entretanto, após a análise dos dados coletados para cada uma das características de software, apenas as 11 primeiras versões do sistema foram consideradas. A análise dos dados consistiu em analisar o desvio padrão de forma cumulativa de cada característica para cada versão do sistema, ou seja, o cálculo do desvio padrão de cada versão considera

os valores das versões anteriores. Essa análise foi importante para definir o menor conjunto de versões necessárias para calibrar o modelo, de forma a obter uma maior precisão no procedimento de simulação. A quantidade de versões que apresentam uma estabilidade desses desvios foi considerada o conjunto mínimo para iniciar o procedimento de simulação.

A Tabela 4-4 apresenta os dados coletados por ARAÚJO (2009) do sistema em questão. O Tamanho é medido em milhares de linhas de código fonte, a Periodicidade é medida pelo intervalo de dias para entrega de uma nova versão, a Complexidade é medida através de complexidade ciclomática média dos métodos das classes, o Esforço é medido pelo número de homens/hora alocados, a Confiabilidade pelo número de defeitos apresentados e, finalmente, a Manutenibilidade, pelo esforço medido em horas, depreendido na correção de defeitos.

Tabela 4-4 – Dados coletados do sistema em observação (ARAÚJO 2009).

#	Tamanho	Periodicidade	Complexidade	Esforço	Confiabilidade	Manutenibilidade
1	62,18	0,00	11,34	6,00	1,00	0,00
2	62,58	1,00	11,41	0,25	1,00	0,25
3	62,52	6,00	11,39	30,75	7,00	30,75
4	62,53	2,00	11,39	3,75	2,00	3,75
5	62,55	7,00	11,39	1,00	1,00	0,00
6	62,55	1,00	11,39	14,00	2,00	14,00
7	62,58	5,00	11,40	2,00	2,00	1,00
8	62,57	1,00	11,40	4,00	3,00	4,00
9	62,57	1,00	11,40	12,00	3,00	9,00
10	62,61	7,00	11,40	6,00	3,00	3,00
11	63,30	4,00	11,43	0,50	1,00	0,50

Após a análise individual de cada versão sobre o comportamento das características de software, foram apresentadas as tendências para cada uma delas (ARAÚJO 2009), assim como apresentado na Tabela 4-5, identificadas a partir da construção da reta de mínimos quadrados. O comportamento individual das características foi apresentado como comportamento esperado.

De acordo com o MPS.BR (2009), a análise de séries históricas permite a identificação de tendências, que podem ser consideradas como um indicador, que é uma estimativa ou avaliação que provê uma base para a tomada de decisão.

Tabela 4-5 – Comportamento das Características de Software (ARAÚJO 2009).

Comportamento	Tamanho	Periodicidade	Complexidade	Esforço	Confiabilidade	Manutenibilidade
Esperado	↑	↑	↑	↓	↓	↑

Com base no comportamento esperado e com as formulações lógicas (ARAÚJO 2009) definidas para cada Lei de Evolução de Software, as seguintes situações podem ser observadas:

- Mudança Contínua = ¬↑ Periodicidade Λ ¬↓ Esforço ⇒ Implicação Negativa
- Incremento da Complexidade = ↑Tamanho v ↑Complexidade v ↑Esforço v ↓ Manutenibilidade ⇒ Implicação Positiva
- Autorregulação = ¬↑ Tamanho Λ ¬↓ Confiabilidade Λ ¬↓ Esforço ⇒ Implicação Negativa
- Conservação da Estabilidade Organizacional = ↔ Esforço ⇒ Implicação Negativa
- Conservação da Familiaridade = ↔ Tamanho Λ ↔ Complexidade Λ ↔ Esforço ⇒ Implicação Negativa
- Crescimento Contínuo = ↑ Tamanho Λ ¬↑ Periodicidade ⇒ Implicação Negativa
- Declínio da Qualidade = ↑ Complexidade v ↑ Esforço v ↓ Confiabilidade v ↓ Manutenibilidade ⇒ Implicação Positiva

4.2.2 Execução da Simulação (Primeira Coleta de Dados)

Para executar a simulação com o novo modelo de observação de evolução de software foi necessário construir as equações que representam cada um dos relacionamentos contidos no modelo. Para isso a planilha construída por ARAÚJO (2009) foi modificada de acordo com as alterações realizadas no modelo, apresentadas no contexto deste trabalho. As alterações consistiram na inclusão do novo relacionamento entre Tamanho e Manutenibilidade, inclusão do relacionamento entre Esforço e Manutenibilidade, que estava ausente na modelagem de Dinâmica de Sistemas e também as alterações realizadas na lógica do cálculo das taxas de cada característica, como visto anteriormente.

Os dados apresentados na Tabela 4-4 são usados como entrada nessa planilha. As equações são construídas através de regressão linear, aplicando-se o método de mínimos quadrados para identificar a tendência do comportamento de uma característica. De acordo com ARAÚJO (2009), é importante ressaltar que o objetivo da observação da evolução de software é visualizar as tendências das curvas, numa visão semiquantitativa, sendo a inclinação da curva gerada por regressão linear apropriada para a análise.

O novo modelo de Dinâmica de Sistemas, apresentado na Tabela 4-3, gerado pela planilha modificada a partir dos dados anteriormente discutidos foi utilizado para a execução da simulação.

A partir da construção do novo modelo de Dinâmica de Sistemas, a simulação é executada e os resultados são apresentados nas figuras Figura 4-5 Figura 4-6. A Figura 4-5 exibe os resultados da simulação, apresentados pela ferramenta *Illium Software Evolution,* através de um gráfico de tendências e através de semáforos indicando o comportamento do sistema para cada Lei de Evolução de Software. De acordo com ARAÚJO (2009), o gráfico não apresenta unidade no eixo y, pois as curvas traçadas para as características estão em unidades distintas, sendo que o objetivo do gráfico é mostrar o comportamento de cada uma delas em uma análise semiquantitativa. Um semáforo é associado a cada Lei de Evolução de Software, sendo que a cor verde significa que a respectiva Lei alcançou sua implicação positiva, a vermelha, a implicação negativa, e a branca, a implicação neutra. A Figura 4-6 exibe os dados gerados pela simulação.

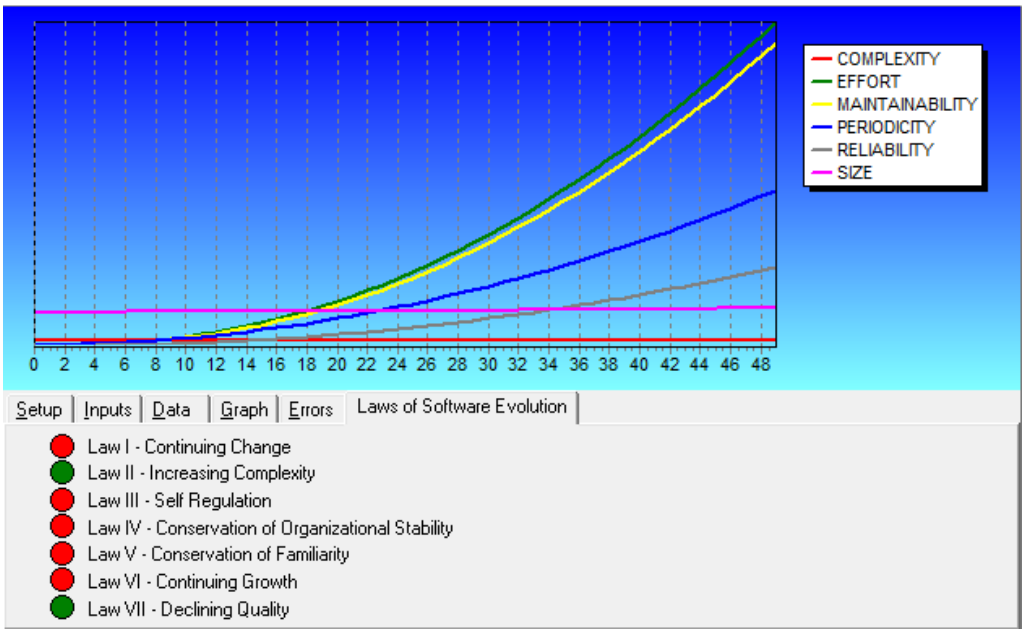


Figura 4-5 – Resultados da Simulação.

	0.0000	1.0000	2.0000	3.0000	4.0000	5.0000	6.0000	7.0000	8.0000	9.0000	10.0000	11.0000	12.0000	13.0000	14.0000	15.0000
COMPLEXI	11.4297	11.4307	11.4326	11.4356	11.4395	11.4444	11.4503	11.4572	11.4651	11.4739	11.4837	11.4945	11.5063	11.5191	11.5329	11.5476
EFFORT	0.5000	1.5000	2.4006	3.3542	4.4784	5.8638	7.5802	9.6814	12.2090	15.1951	18.6641	22.6352	27.1230	32.1389	37.6915	43.7875
MAINTAINA	0.5000	1.4740	2.3103	3.1628	4.1506	5.3653	6.8774	8.7415	10.9995	13.6837	16.8192	20.4252	24.5164	29.1044	34.1979	39.8037
PERIODICI	4.0000	4.2273	4.6818	5.3636	6.2727	7.4091	8.7727	10.3636	12.1818	14.2273	16.5000	19.0000	21.7273	24.6818	27.8636	31.2727
RELIABILIT	1.0000	1.2055	1.4299	1.6992	2.0333	2.4475	2.9537	3.5611	4.2766	5.1057	6.0526	7.1205	8.3119	9.6287	11.0725	12.6442

Figura 4-6 – Dados gerados pela simulação.

A Tabela 4-6 apresenta uma comparação entre as tendências do comportamento esperado para o sistema, apresentadas anteriormente, e as tendências do comportamento obtidas através dos resultados da simulação.

Tabela 4-6 – Comparação entre os comportamentos esperados e simulados.

Comportamento	Tamanho	Periodicidade	Complexidade	Esforço	Confiabilidade	Manutenibilidade
Esperado	↑	↑	↑	↓	↓	↑
Simulado	↑	↑	↑	↑	↓	↓

O comportamento esperado para as características de software representa a situação real do sistema em um determinado período do tempo, considerando que o comportamento do sistema se mantenha nas versões seguintes (ARAÚJO 2009).

Com os resultados da simulação pode-se perceber que o comportamento esperado das características Esforço e Manutenibilidade não correspondem ao comportamento simulado. A diferença entre o comportamento esperado e simulado das características pode ser explicada pelo fato do cálculo do comportamento de cada característica, realizado através da simulação, considerar a influência entre as características e não apenas uma característica isoladamente, como é feito no cálculo do comportamento esperado. No caso da característica Esforço, no novo modelo ela é influenciada pelas características Tamanho, Periodicidade, Complexidade e Confiabilidade. E no caso da Manutenibilidade, no novo modelo ela é influenciada pelas características Tamanho, Esforço e Complexidade.

A diferença no resultado da característica Esforço já tinha sido observada anteriormente por ARAÚJO (2009) e, portanto, não é uma consequência das alterações realizadas no modelo, já a alteração no resultado da característica Manutenibilidade pode ser uma consequência das alterações realizadas no modelo.

Para verificar a coerência dos resultados do comportamento simulado é necessário coletar dados de versões futuras do sistema.

4.2.3 Situação Real do Sistema x Resultados da Simulação (Primeira Coleta de Dados)

Para analisar os resultados da simulação apresentados na seção anterior serão apresentados os dados coletados de versões posteriores às coletadas para a construção das equações de simulação. Assim, será possível verificar a situação real do sistema e comparar com os resultados da simulação. Novamente, os dados utilizados são os mesmos apresentados por ARAÚJO (2009), para que seja possível

realizar uma comparação entre os resultados da simulação utilizando o modelo original e o novo modelo no contexto desse trabalho.

A Tabela 4-7 apresenta dados de mais oito versões do sistema em observação, posteriores às coletadas para a construção das equações de simulação. Para fins de comparação de resultados esse dados foram os mesmos utilizados por ARAÚJO (2009). As unidades de cada medida são as mesmas utilizadas na coleta anterior apresentada na Tabela 4-4.

Tabela 4-7 – Dados coletados de novas versões do sistema (ARAÚJO 2009).

#	Tamanho	Periodicidade	Complexidade	Esforço	Confiabilidade	Manutenibilidade
1	63,99	14,00	11,57	4,75	1,00	4,75
2	64,31	8,00	11,63	22,50	4,00	2,50
3	64,31	20,00	11,63	2,00	1,00	2,00
4	64,37	7,00	11,64	29,50	8,00	13,50
5	65,43	30,00	11,85	53,50	8,00	6,50
6	65,76	5,00	11,93	10,50	5,00	3,50
7	65,96	16,00	11,96	3,50	1,00	3,50
8	65,96	48,00	11,96	25,75	5,00	3,00

Após a análise individual de cada versão sobre comportamento das características de software, foram apresentadas as tendências para cada uma delas (ARAÚJO 2009), assim como apresentado na Tabela 4-8. O comportamento individual das características é apresentado como comportamento observado, representado a situação real do sistema.

Com os dados coletados da situação real do sistema, tanto de um momento anterior (comportamento esperado) quanto de um momento posterior (comportamento observado) ao procedimento de simulação, foi possível avaliar os resultados obtidos pela simulação (comportamento simulado). A Tabela 4-8 apresenta um comparativo entre o comportamento esperado, calculado com dos dados das versões utilizadas para construção das equações do modelo, o comportamento simulado e o comportamento observado, calculado com dados de novas versões do sistema.

Tabela 4-8– Comparação entre os comportamentos Esperado, Simulado e Observado.

Comportamento	Tamanho	Periodicidade	Complexidade	Esforço	Confiabilidade	Manutenibilidade
Esperado	↑	↑	↑	↓	↓	↑
Simulado	↑	↑	↑	↑	↓	↓
Observado	↑	↑	↑	↑	↓	↑

Como se pode perceber, o comportamento observado do sistema não corresponde exatamente aos comportamentos simulados com o novo modelo para todas as características analisadas. Apesar do comportamento esperado para o Esforço apresentar uma tendência de diminuição, a simulação mostrou uma tendência

contrária, que foi confirmada pelo comportamento observado do sistema. Por outro lado, o comportamento simulado para a característica Manutenibilidade não foi confirmado pelo comportamento observado. A princípio esse resultado pode parecer insatisfatório, já que os comportamentos simulados não foram consistentes com os comportamentos observados. Entretanto, esse resultado deve ser analisado detalhadamente.

Como dito anteriormente, a característica Manutenibilidade foi medida através do total de horas investidas na correção de defeitos e a característica Confiabilidade foi medida através da quantidade de defeitos encontrados no sistema. Como se pode observar na Tabela 4-8, o comportamento para confiabilidade tende a ser de diminuição, ou seja, o número de defeitos tende a aumentar. Portanto, com o número de defeitos aumentando, a tendência da Manutenibilidade pode diminuir, já que devido à maior quantidade de defeitos o tempo gasto na correção pode aumentar.

O comportamento observado mostra que mesmo com o número de defeitos aumentando, o esforço gasto para corrigir os defeitos diminuiu, ou seja, a Confiabilidade no sistema tende a diminuir e a Manutenibilidade tende a aumentar. Analisando o contexto do momento da coleta dos dados, esse resultado pode ser explicado pelo fato de nesse período o sistema estar em uma fase inicial do desenvolvimento onde possivelmente grande parte do esforço foi investido na construção de novas funcionalidades. É importante ressaltar que esses resultados são específicos para o conjunto de dados utilizados no estudo e, portanto, não podem ser generalizados.

4.3 Modelo Original x Novo Modelo

Para destacar o impacto das alterações realizadas tanto no modelo de observação da evolução de software conceitual quanto no modelo de Dinâmica de Sistemas, esta seção apresenta uma comparação entre os resultados da simulação do modelo original de Dinâmica de Sistemas (ARAÚJO 2009), apresentado na coluna esquerda da Tabela 4-3, e os resultados da simulação do novo modelo de Dinâmica de Sistemas, apresentado na coluna direita da Tabela 4-3.

Embora a análise realizada sobre os resultados das simulações seja semiquantitativa, onde o importante é a tendência dos dados e não seus valores, a Tabela 4-9 apresenta os valores dos resultados obtidos por ARAÚJO (2009) e os resultados obtidos com a simulação do novo modelo apresentada neste trabalho. O objetivo é apenas ilustrar a diferença dos resultados e possibilitar uma comparação mais detalhada. A comparação das tendências será apresentada logo em seguida. É importante ressaltar que as duas simulações foram executadas utilizando o mesmo

conjunto de dados e que a única diferença na execução da simulação está no modelo de simulação utilizado. As simulações foram executadas utilizando a ferramenta *Illium Software Evolution* (ARAÚJO 2009).

Tabela 4-9 – Comparação entre os resultados da simulação: Modelo Original (O) x Novo Modelo (N).

Tempo	Tamanho		Periodicidade		Complexidade		Esforço		Confiabilidade		Manutenibilidade	
	O	N	O	N	O	N	O	N	O	N	O	N
0	63,29	63,29	4,00	4,00	11,43	11,43	0,50	0,50	1,00	1,00	0,50	0,50
1	63,29	63,29	4,00	4,00	11,43	11,43	0,50	1,50	1,00	1,17	0,50	1,51
2	63,30	63,30	4,23	4,23	11,43	11,43	0,65	2,27	1,04	1,34	0,50	2,25
3	63,31	63,31	4,68	4,68	11,43	11,43	0,95	2,99	1,12	1,53	0,50	2,91
4	63,33	63,33	5,36	5,36	11,43	11,43	1,39	3,81	1,24	1,78	0,50	3,63
5	63,35	63,35	6,28	6,28	11,44	11,44	1,99	4,83	1,40	2,09	0,50	4,51
6	63,38	63,38	7,41	7,41	11,44	11,44	2,74	6,14	1,60	2,49	0,50	5,64
7	63,42	63,42	8,77	8,77	11,45	11,45	3,63	7,79	1,84	2,99	0,50	7,09
8	63,46	63,46	10,36	10,36	11,46	11,46	4,68	9,84	2,12	3,59	0,50	8,91
9	63,51	63,51	12,18	12,18	11,46	11,46	5,87	12,33	2,44	4,29	0,50	11,13
10	63,56	63,56	14,23	14,23	11,47	11,47	7,21	15,29	2,80	5,12	0,50	13,78
11	63,62	63,62	16,50	16,50	11,48	11,48	8,70	18,74	3,21	6,06	0,50	16,89
12	63,68	63,68	19,00	19,00	11,49	11,49	10,34	22,69	3,65	7,13	0,50	20,48
13	63,75	63,75	21,73	21,73	11,51	11,51	12,13	27,17	4,13	8,32	0,50	24,56
14	63,83	63,83	24,68	24,68	11,52	11,52	14,07	32,17	4,65	9,63	0,50	29,14
15	63,91	63,91	27,86	27,86	11,53	11,53	16,16	37,72	5,21	11,08	0,50	34,22
16	63,99	63,99	31,27	31,27	11,55	11,55	18,39	43,81	5,81	12,65	0,50	39,82
17	64,09	64,09	34,91	34,91	11,56	11,56	20,78	50,45	6,46	14,35	0,50	45,94
18	64,19	64,19	38,77	38,77	11,58	11,58	23,32	57,64	7,14	16,18	0,50	52,58
19	64,29	64,29	42,86	42,86	11,59	11,59	26,00	65,39	7,86	18,14	0,50	59,75
20	64,40	64,40	47,18	47,18	11,62	11,62	28,84	73,70	8,62	20,23	0,50	67,45

A partir dos dados apresentados pode-se observar que os resultados para as características Tamanho, Periodicidade e Complexidade foram os mesmos, utilizando os diferentes modelos. Para as características Esforço, Confiabilidade e Manutenibilidade os valores resultados obtidos com cada modelo foram diferentes. As tendências para Esforço e Confiabilidade também foram mantidas, entretanto, a tendência para Manutenibilidade foi alterada. Isso pode ser explicado pelo fato de que as principais alterações no modelo estão relacionadas às características Esforço, Confiabilidade e Manutenibilidade, que formam uma referência circular (*loop*) de influências e passaram a ser consideradas na modelagem de Dinâmica de Sistemas. A alteração no modelo relacionada à inclusão do novo relacionamento entre as características Tamanho e Manutenibilidade não afetou a característica Tamanho, pois esse relacionamento significa que o Tamanho influencia na Manutenibilidade, dessa forma, neste caso, apenas a Manutenibilidade é afetada.

Para explicitar as diferenças entre os resultados da simulação com o modelo original e com o novo modelo, as figuras de 4-7 a 4-12 mostram os gráficos gerados com os dados apresentados na Tabela 4-9.

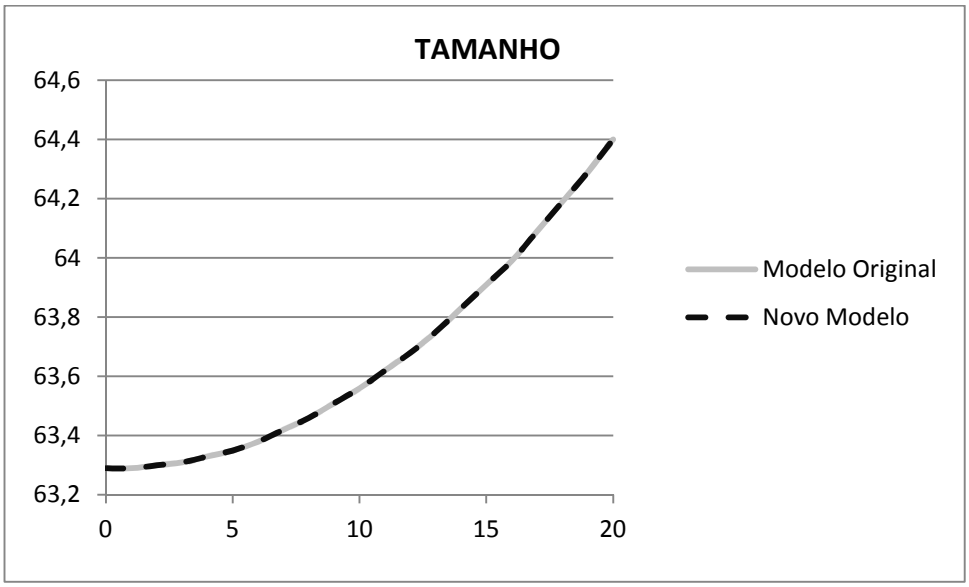


Figura 4-7 – Resultados da simulação para a característica Tamanho.

De acordo com a Figura 4-7, o modelo original e o novo modelo apresentam os mesmos resultados para a característica Tamanho.

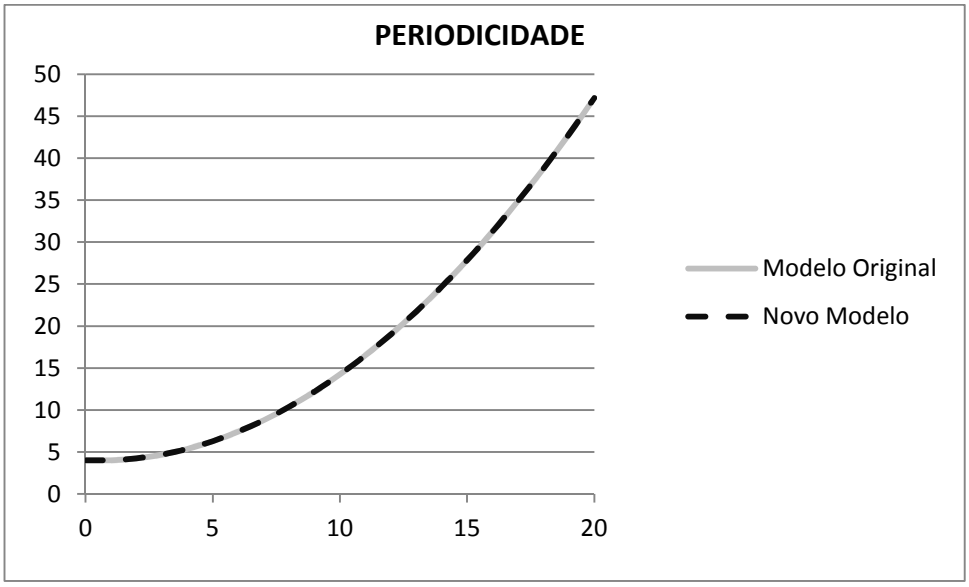


Figura 4-8 – Resultados da simulação para a característica Periodicidade.

De acordo com a Figura 4-8, o modelo original e o novo modelo apresentam os mesmos resultados para a característica Periodicidade.

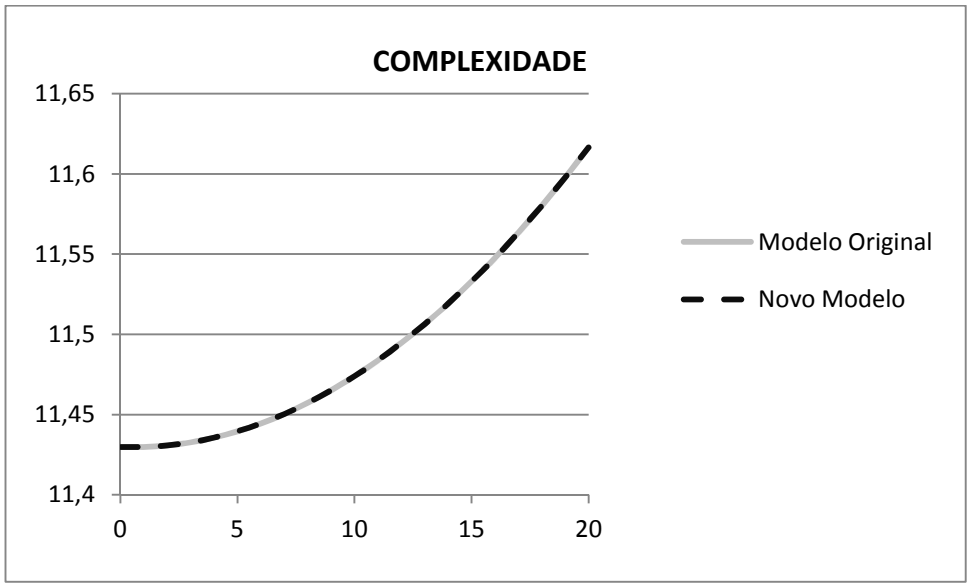


Figura 4-9 – Resultados da simulação para a característica Complexidade.

De acordo com a Figura 4-9, o modelo original e o novo modelo apresentam os mesmos resultados para a característica Complexidade.

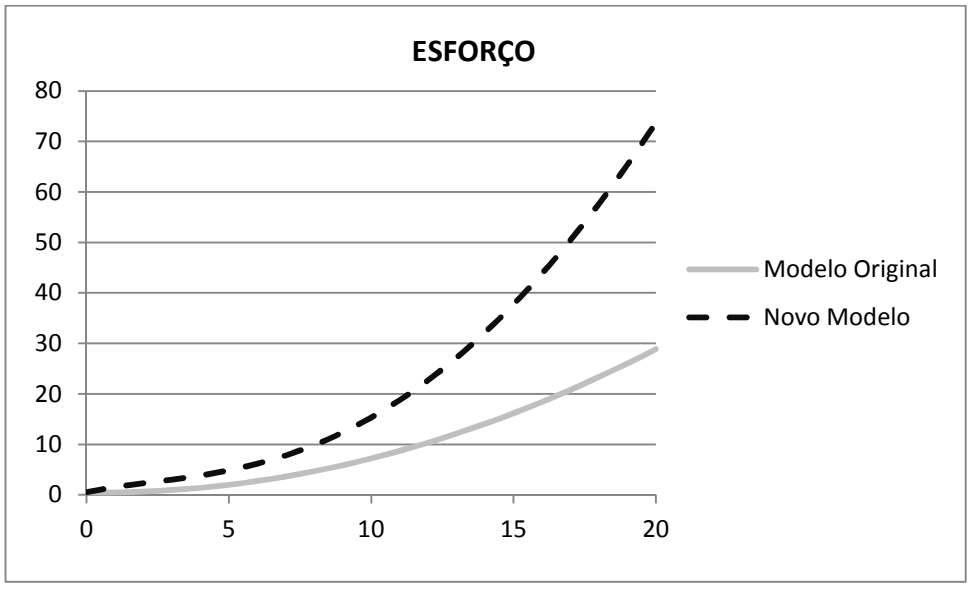


Figura 4-10 – Resultados da simulação para a característica Esforço.

De acordo com a Figura 4-10, o modelo original e o novo modelo apresentam resultados diferentes para a característica Esforço, apesar de apresentarem a mesma tendência. A diferença nos resultados pode ser explicada pela inclusão do relacionamento entre Esforço e Manutenibilidade no modelo de Dinâmica de Sistemas, que é responsável por formar o *loop* de realimentação entre as variáveis Esforço, Manutenibilidade e Confiabilidade. Nesse caso, a variável Esforço é influenciada pela variável Confiabilidade, que é influenciada pela variável Manutenibilidade, que influencia o Esforço.

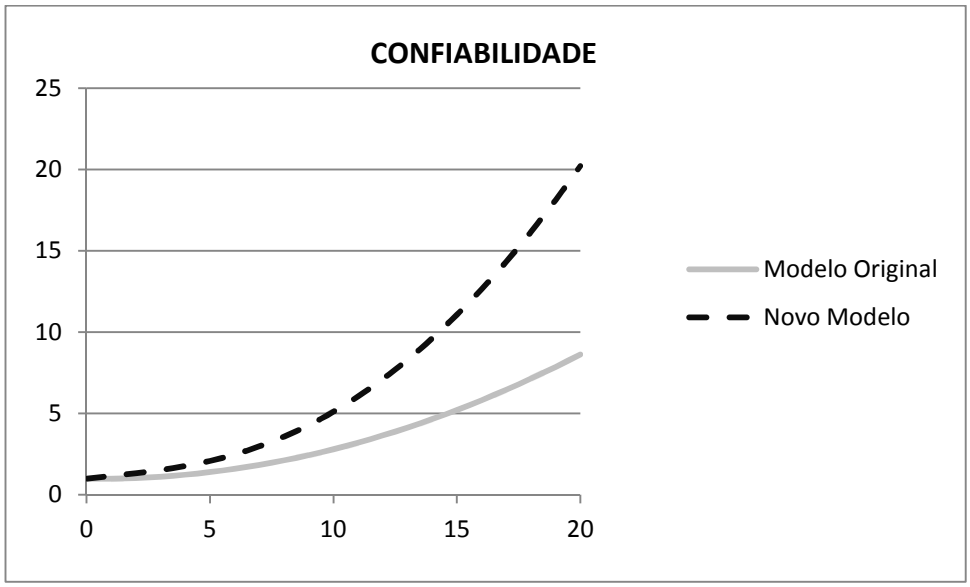


Figura 4-11 – Resultados da simulação para a característica Confiabilidade.

De acordo com a Figura 4-11, o modelo original e o novo modelo apresentam resultados diferentes para a característica Confiabilidade, apesar de apresentarem a mesma tendência. A diferença nos resultados pode ser explicada pela inclusão do relacionamento entre Esforço e Manutenibilidade no modelo de Dinâmica de Sistemas, que é responsável por formar o *loop* de realimentação entre as variáveis Esforço, Manutenibilidade e Confiabilidade.

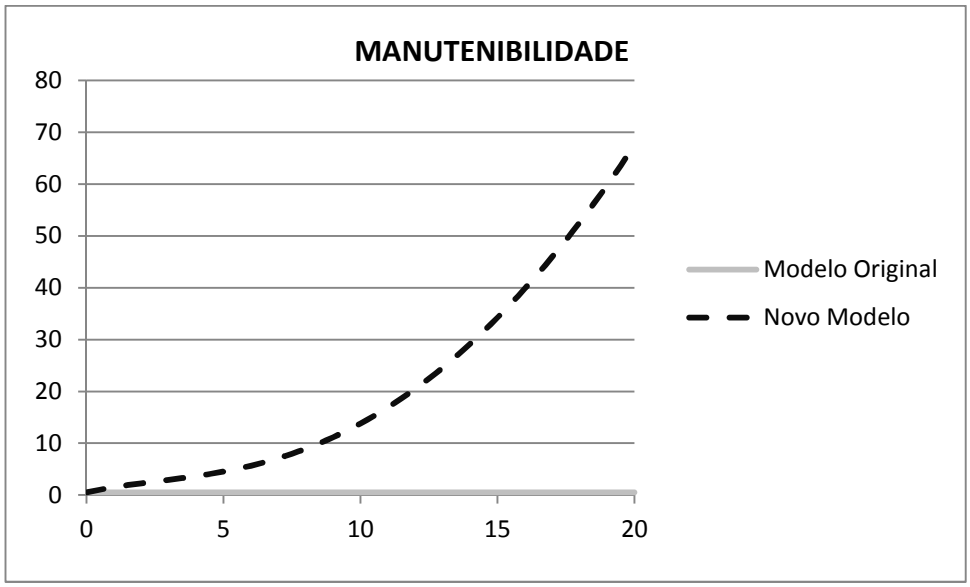


Figura 4-12 – Resultados da simulação para a característica Manutenibilidade.

De acordo com a Figura 4-12, o modelo original e o novo modelo apresentam resultados diferentes para a característica Manutenibilidade. A diferença nos resultados pode ser explicada pela inclusão do relacionamento entre Esforço e

Manutenibilidade no modelo de Dinâmica de Sistemas, ou seja, o cálculo da variável Manutenibilidade passou a considerar a variável Esforço.

Na Tabela 4-10, as três primeiras linhas apresentam os resultados obtidos por ARAÚJO (2009) na forma de tendências de comportamento para cada característica de software. A primeira linha apresenta o comportamento esperado, baseado nas tendências das medidas coletadas e considerando que o comportamento do sistema se mantenha nas versões seguintes; a segunda linha apresenta o comportamento observado baseado nas tendências das medidas coletadas de versões posteriores às versões utilizadas na primeira linha; a terceira linha apresenta o comportamento do sistema obtido com os resultados da simulação do modelo original (ARAÚJO, 2009); e a quarta linha apresenta o comportamento do sistema obtido com os resultados da simulação do novo modelo. A coluna destacada ressalta a diferença entre os resultados obtidos com o modelo original e com o novo modelo.

Tabela 4-10 – Comparativo entre os comportamentos Esperado, Observado, Simulado (Modelo Original) e Simulado (Novo Modelo).

Comportamento	Tamanho	Periodicidade	Complexidade	Esforço	Confiabilidade	Manutenibilidade
Esperado	↑	↑	↑	↓	↓	↑
Observado	↑	↑	↑	↑	↓	↑
Simulado Modelo Original	↑	↑	↑	↑	↓	↔
Simulado Novo Modelo	↑	↑	↑	↑	↓	↓

Os resultados apresentados permitem concluir que as alterações realizadas no modelo causaram um impacto significativo nos resultados das tendências para o conjunto de dados utilizados no estudo.

Na simulação com modelo original a tendência da característica Manutenibilidade se manteve constante devido à restrição imposta no modelo de Dinâmica de Sistema, através da funcionalidade MAX, que impedia que os valores das características diminuíssem com o tempo, mesmo se o coeficiente fosse negativo, já que, no modelo original o termo que calcula a variação da característica Manutenibilidade é negativo (consulte a Tabela 4-1 coluna do “Modelo Original”).

Além da alteração da tendência de uma das características, o valor das características Esforço e Confiabilidade foram alterados consideravelmente, passando a ter um valor três vezes superior àquele obtido anteriormente com a simulação do modelo original, o que pode significar uma situação de extremo risco para o projeto. Embora neste trabalho o foco seja a análise das tendências, esse resultado pode ser explorado em trabalhos futuros.

É importante ressaltar que o relacionamento entre Esforço e Manutenibilidade forma um *loop* de relacionamento entre as características Esforço, Manutenibilidade e Confiabilidade, e por esse motivo causa um impacto significativo nos resultados dessas características na simulação com o novo modelo.

4.4 Simulação com o Novo Modelo (Segunda Coleta de Dados)

Para executar um novo estudo de simulação foram coletados novos dados do mesmo sistema apresentado anteriormente. A nova coleta foi identificada como segunda coleta de dados.

Os resultados obtidos com esse novo estudo podem aumentar a confiança no novo modelo de simulação apresentado neste trabalho. Entretanto, a validade de generalização dos resultados obtidos com a simulação do novo modelo não pode ser garantida, já que apenas um sistema de software está sendo utilizado para avaliar o modelo. Além disso, os resultados deste novo estudo permitem dar continuidade à observação da evolução do sistema em questão, possibilitando assim, futuras avaliações do modelo e também maior apoio à equipe de desenvolvimento do projeto na tomada de decisões com base em um maior entendimento sobre o processo de evolução do sistema.

4.4.1 Análise dos Novos Dados

Os novos dados foram coletados de versões posteriores àquelas já apresentadas anteriormente pela primeira coleta de dados. Entretanto, eles não foram adicionados ao conjunto dos dados originais, pois a janela de tempo entre a primeira e a segunda coleta foi grande (aproximadamente seis meses), portanto, a última versão da primeira coleta não é imediatamente posterior à primeira versão da segunda coleta.

Foram coletados dados de 6 novas versões do sistema em questão. A etapa do desenvolvimento de software observada no novo estudo também foi a de Codificação. Para isso, as métricas listadas na Tabela 4-11, relacionadas ao código fonte do sistema, foram analisadas para realização da coleta dos dados.

Tabela 4-11 – Métricas utilizadas para cada característica de software.

Característica	Métrica
Tamanho	Milhares de linhas de código.
Periodicidade	Dias entre as datas de liberação do sistema.
Complexidade	Complexidade Ciclomática média do código multiplicada pela quantidade de métodos.
Esforço	Tempo (em horas) gasto com atividades do desenvolvimento.
Confiabilidade	Número de defeitos reportados em cada versão.
Manutenibilidade	Tempo (em horas) gasto na correção dos defeitos reportados.

As métricas para as características Tamanho e Complexidade podem ser calculadas com o auxilio de ferramentas de coleta de dados automática em código fonte. A métrica para Periodicidade é calculada a partir das datas da liberação de cada versão do sistema para o cliente. A métrica para Esforço é medida pela quantidade

total de tempo gasto com atividades do desenvolvimento relacionadas à codificação. A métrica para Confiabilidade é calculada a partir dos registros de defeitos detectados durantes seções de teste beta realizadas com os usuários do sistema. A métrica para Manutenibilidade é calculada através do tempo gasto para a correção dos defeitos detectados. Essas informações são extraídas dos registros realizados pelos desenvolvedores na ferramenta de acompanhamento do desenvolvimento utilizada no projeto.

Para garantir uma padronização dos dados coletados diretamente a partir do código fonte, ele foi formatado automaticamente de acordo com um padrão de codificação criado pela equipe de desenvolvimento do projeto analisado. É importante ressaltar que nesse período o desenvolvimento estava passando por uma fase de intensa atividade de testes beta, o que resultou em mais atividades de manutenção.

A Tabela 4-12 apresenta os novos dados coletados do sistema em questão. As métricas para cada característica correspondem às apresentadas na Tabela 4-11.

Tabela 4-12 – Novos dados do sistema em observação.

#	Tamanho	Periodicidade	Complexidade	Esforço	Confiabilidade	Manutenibilidade
1	182,39	0,00	35,40	31,42	7,00	20,00
2	182,94	3,00	35,54	62,72	1,00	18,00
3	183,21	2,00	35,55	42,28	7,00	14,00
4	183,40	3,00	35,57	46,75	2,00	16,00
5	184,31	7,00	35,85	61,75	8,00	24,00
6	184,21	16,00	36,16	82,22	4,00	26,00

A Tabela 4-13 apresenta o desvio padrão calculado de forma cumulativa para cada versão do sistema, a partir da primeira versão. De acordo com ARAÚJO (2009), isso é importante para ajudar a definir o menor número de versões que precisam ser utilizadas na calibração do modelo, de forma a ter a maior precisão no procedimento de simulação.

Tabela 4-13 – Desvio padrão cumulativo

#	Tamanho	Periodicidade	Complexidade	Esforço	Confiabilidade	Manutenibilidade
1						
2	0,39	2,12	0,09	22,13	4,24	1,41
3	0,42	1,52	0,08	15,89	3,46	3,05
4	0,43	1,41	0,07	12,99	3,46	2,58
5	0,69	2,54	0,16	13,32	3,49	3,84
6	0,73	5,77	0,27	18,05	3,14	4,63

Devido a certa estabilidade apresentada pelo desvio padrão das características entre as seis versões analisadas do sistema, todas as versões foram utilizadas para gerar as equações do modelo de simulação.

Para analisar a situação real do sistema, no período correspondente às versões analisadas, as figuras de 4-13 a 4-18 apresentam um gráfico da tendência

para analisar o comportamento individual de cada uma das características de software analisadas. As retas traçadas nos gráficos foram calculadas através de regressão linear.

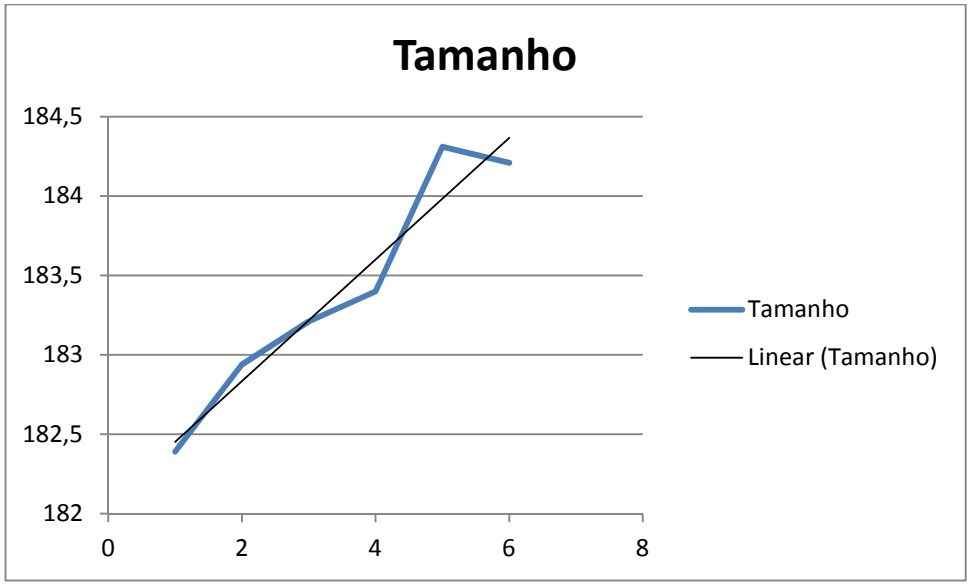


Figura 4-13 – Comportamento da característica Tamanho.

A Figura 4-13 apresenta um comportamento de tendência da característica Tamanho, observando os dados históricos do sistema em observação.

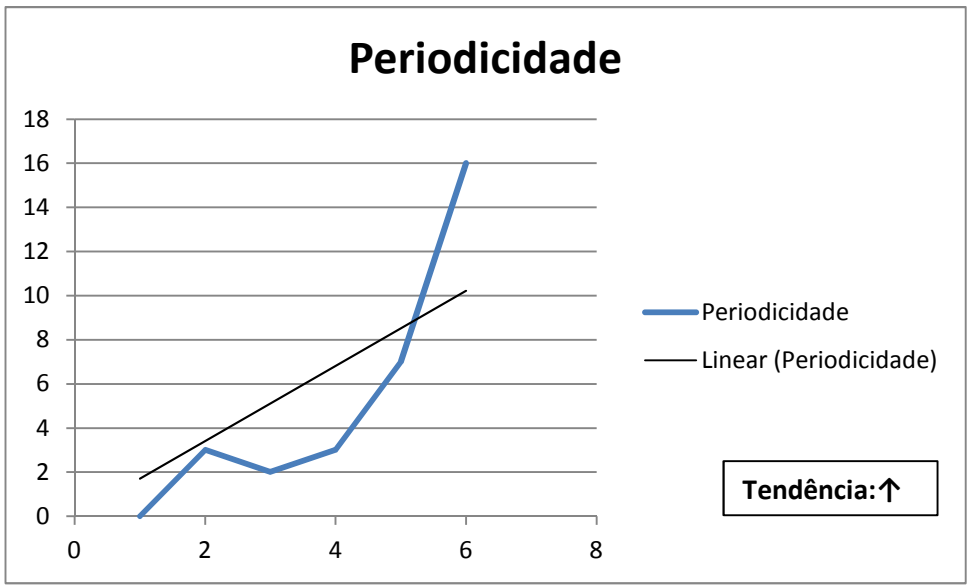


Figura 4-14 – Comportamento da característica Periodicidade.

A Figura 4-14 apresenta um comportamento de tendência da característica Periodicidade, observando os dados históricos do sistema em observação.

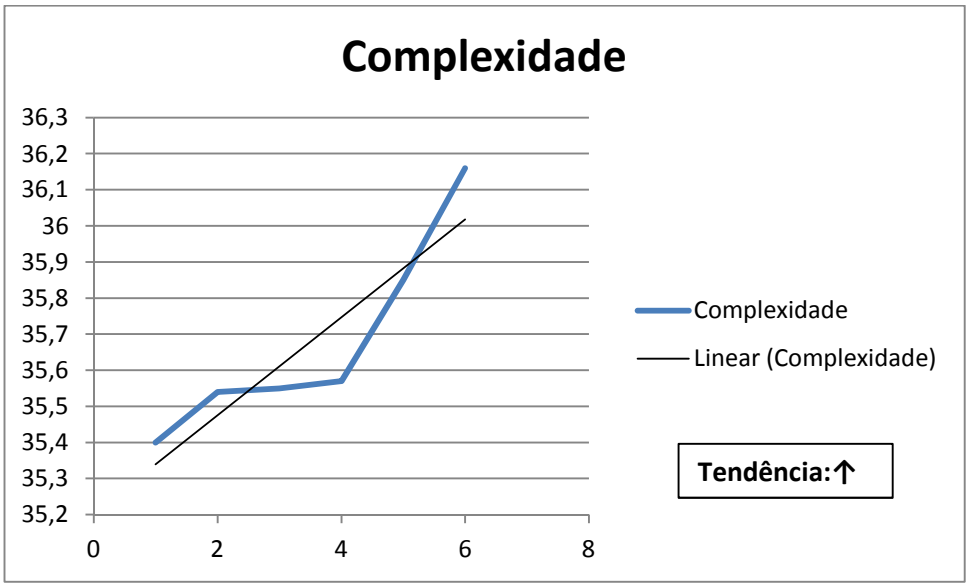


Figura 4-15 – Comportamento da característica Complexidade.

A Figura 4-15 apresenta um comportamento de tendência da característica Complexidade, observando os dados históricos do sistema em observação.

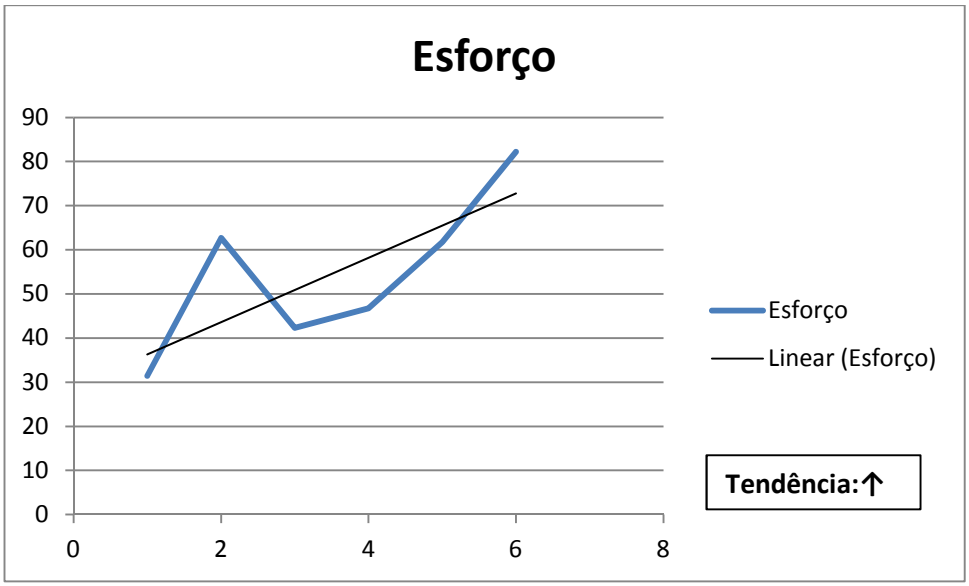


Figura 4-16 – Comportamento da característica Esforço.

A Figura 4-16 apresenta um comportamento de tendência da característica Esforço, observando os dados históricos do sistema em observação.

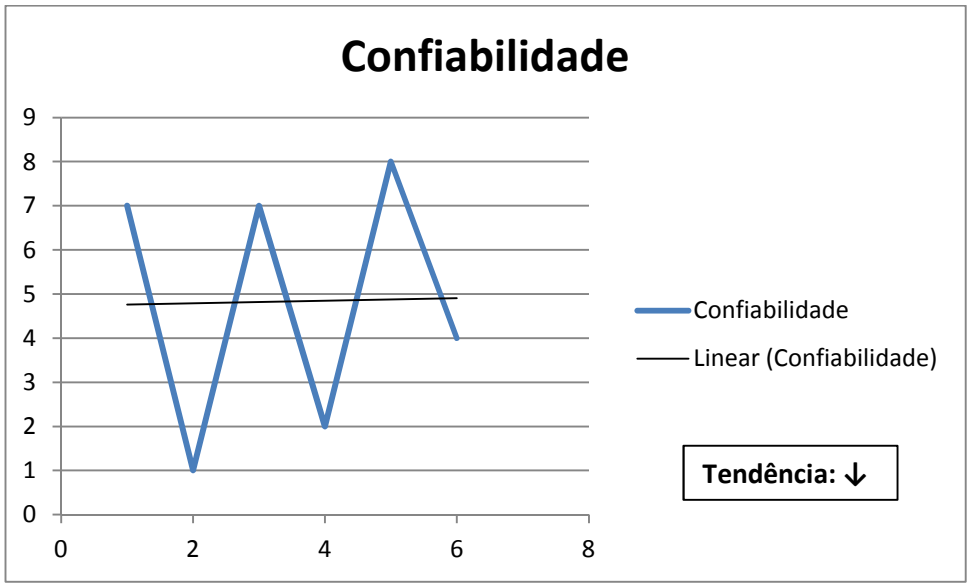


Figura 4-17 – Comportamento da característica Confiabilidade.

A Figura 4-17 apresenta um comportamento de tendência da característica Confiabilidade, observando os dados históricos do sistema em observação. É importante ressaltar que a Confiabilidade foi medida através do número de defeitos encontrados em cada versão do sistema, e que quanto maior a quantidade de defeitos menor é a Confiabilidade.

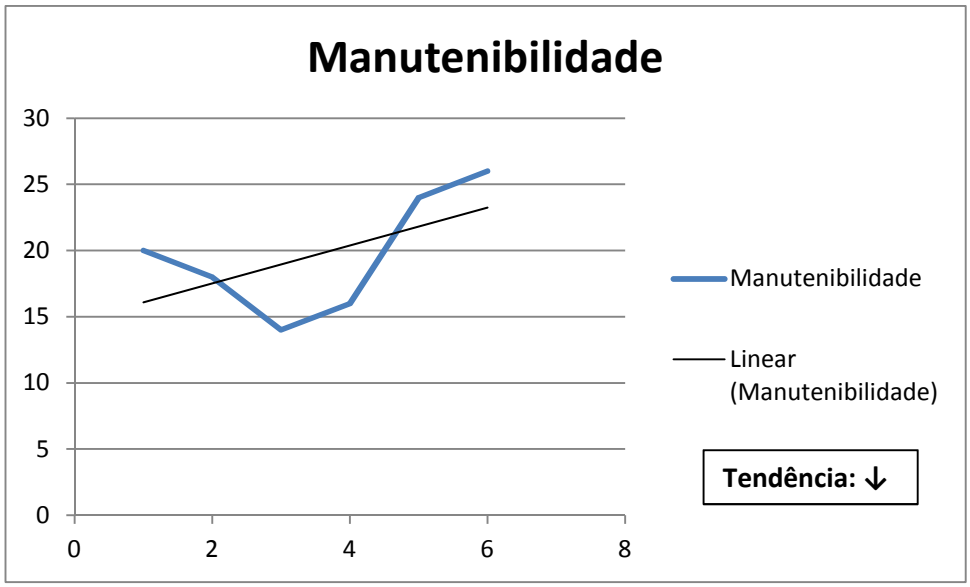


Figura 4-18 – Comportamento da característica Manutenibilidade.

A Figura 4-18 apresenta um comportamento de tendência da característica Manutenibilidade, observando os dados históricos do sistema em observação. A Manutenibilidade foi medida através do tempo gasto na correção dos defeitos,

portanto, quanto maior o tempo gasto na correção menor é a Manutenibilidade do sistema.

Após a análise individual de cada versão sobre comportamento das características de software, a Tabela 4-14 apresenta as tendências para cada uma delas. O comportamento individual das características é apresentado como comportamento esperado.

Tabela 4-14 – Comportamento esperado para novos dados do sistema em observação.

Comportamento	Tamanho	Periodicidade	Complexidade	Esforço	Confiabilidade	Manutenibilidade
Esperado	↑	↑	↑	↑	↓	↓

Com base no comportamento esperado e de acordo com as formulações lógicas ARAÚJO (2009) definidas para cada Lei de Evolução de Software, as seguintes situações podem ser observadas para os novos dados:

- Mudança Contínua = $\neg\uparrow$ Periodicidade $\wedge$ $\neg\downarrow$ Esforço $\Rightarrow$ Implicação Negativa
- Incremento da Complexidade = $\uparrow$ Tamanho v $\uparrow$ Complexidade v $\uparrow$ Esforço v $\downarrow$ Manutenibilidade $\Rightarrow$ Implicação Positiva
- Autorregulação = $\neg\uparrow$ Tamanho $\wedge$ $\neg\downarrow$ Confiabilidade $\wedge$ $\neg\downarrow$ Esforço $\Rightarrow$ Implicação Negativa
- Conservação da Estabilidade Organizacional = $\leftrightarrow$ Esforço $\Rightarrow$ Implicação Negativa
- Conservação da Familiaridade = $\leftrightarrow$ Tamanho $\wedge$ $\leftrightarrow$ Complexidade $\wedge$ $\leftrightarrow$ Esforço $\Rightarrow$ Implicação Negativa
- Crescimento Contínuo = $\uparrow$ Tamanho $\wedge$ $\neg\uparrow$ Periodicidade $\Rightarrow$ Implicação Negativa
- Declínio da Qualidade = $\uparrow$ Complexidade v $\uparrow$ Esforço v $\downarrow$ Confiabilidade v $\downarrow$ Manutenibilidade $\Rightarrow$ Implicação Positiva

A análise apresentada nesta seção é uma análise pontual do sistema e realizada de forma independente para cada característica de software e é utilizada para observar o comportamento esperado do sistema.

4.4.2 Execução da Simulação (Segunda Coleta de Dados)

Para executar a simulação do modelo de observação de evolução de software foi necessário construir as equações que representam cada um dos relacionamentos contidos no modelo com os novos dados coletados. Para isso a planilha construída por

ARAÚJO (2009), modificada de acordo com as alterações realizadas no modelo, foi utilizada.

Os novos dados coletados, apresentados na Tabela 4-12, são usados como entrada nessa planilha. É importante ressaltar que o objetivo da observação da evolução de software é visualizar as tendências das curvas, numa visão semiquantitativa, sendo a inclinação da curva gerada por regressão linear apropriada para a análise ARAÚJO (2009).

O modelo de Dinâmica de Sistemas, gerado pela planilha a partir dos dados anteriormente discutidos, é apresentado a seguir utilizando a linguagem de modelagem definida por BARROS *et al.* (2000). As palavras destacadas em negrito representam palavras-chave da linguagem. O modelo apresentado está no formato necessário para executar a simulação na ferramenta *Illium Software Evolution.* É importante ressaltar que o modelo a seguir é o novo modelo de observação de evolução de software, resultado dos estudos realizados neste trabalho.

```
#
######################################################################
SPEC DT 1.00;
######################################################################
# SOFTWARE CHARACTERISTICS
######################################################################

STOCK SIZE 184.21;
STOCK PERIODICITY 16;
STOCK COMPLEXITY 36.16004;
STOCK EFFORT 82.22;
STOCK RELIABILITY 4;
STOCK MAINTAINABILITY 26;
STOCK TIMER 1;

STOCK DELTA_EFFORT 82.22;

######################################################################
# RATES
######################################################################

PROC DELTA_PERIODICITY 2.66 * TIMER;
PROC DELTA_SIZE 0.10 * DELTA_PERIODICITY;
PROC DELTA_COMPLEXITY 0.33 * DELTA_SIZE + 0.05 * DELTA_PERIODICITY;
PROC DELTA_MAINTAINABILITY 13.12 * DELTA_COMPLEXITY + 0.17 *
DELTA_EFFORT + 3.87 * DELTA_SIZE;
PROC DELTA_RELIABILITY 0.49 * DELTA_SIZE + (-0.04 * DELTA_PERIODICITY) +
0.28 * DELTA_COMPLEXITY + 0.13 * DELTA_MAINTAINABILITY;
PROC DELTA_EFFORT_AUX 18.65 * DELTA_SIZE + 2.82 * DELTA_PERIODICITY +
57.82 * DELTA_COMPLEXITY + (-2.01 * DELTA_RELIABILITY);

RATE (SOURCE, TIMER) RTT DT;
RATE (SOURCE, PERIODICITY) RTP IF (AND (PERIODICITY <= 0,
DELTA_PERIODICITY <0), -PERIODICITY, DELTA_PERIODICITY) * DT;
RATE (SOURCE, SIZE) RTS IF (AND (SIZE <= 0, DELTA_SIZE <0), -SIZE,
DELTA_SIZE) * DT;
RATE (SOURCE, COMPLEXITY) RTC IF (AND (COMPLEXITY <= 0,
DELTA_COMPLEXITY <0), -COMPLEXITY, DELTA_COMPLEXITY) * DT;
RATE (SOURCE, MAINTAINABILITY) RTM IF (AND (MAINTAINABILITY <= 0,
DELTA_MAINTAINABILITY <0), -MAINTAINABILITY, DELTA_MAINTAINABILITY) *
DT;
RATE (SOURCE, RELIABILITY) RTR IF (AND (RELIABILITY <= 0,
DELTA_RELIABILITY <0), -RELIABILITY, DELTA_RELIABILITY) * DT;
RATE (SOURCE, EFFORT) RTE IF (AND (EFFORT <= 0, DELTA_EFFORT <0), -
EFFORT, DELTA_EFFORT) * DT;

RATE (SOURCE, DELTA_EFFORT) RTDEO DELTA_EFFORT_AUX -
DELTA_EFFORT;
```

A partir da construção do modelo, a simulação é executada e os resultados são apresentados nas figuras Figura 4-19 e Figura 4-20. A Figura 4-19 exibe os resultados da simulação, apresentados pela ferramenta *Illium Software Evolution,* através de um

gráfico de tendências e através de semáforos indicando o comportamento do sistema para cada Lei de Evolução de Software. A Figura 4-20 apresenta os dados gerados pela simulação.

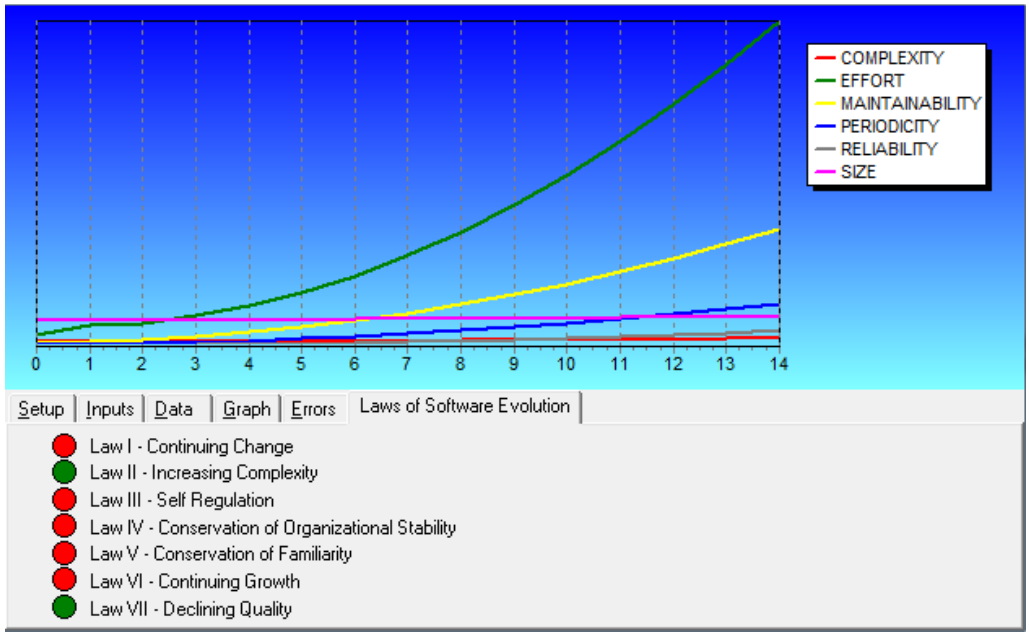


Figura 4-19 – Resultados da Simulação.

	0.0000	1.0000	2.0000	3.0000	4.0000	5.0000	6.0000	7.0000	8.0000	9.0000	10.0000	11.0000	12.0000	13.0000	14.0000
COMPLEXI	36.1600	36.3768	36.8103	37.4605	38.3275	39.4112	40.7117	42.2289	43.9629	45.9136	48.0810	50.4652	53.0662	55.8839	58.9183
EFFORT	82.2200	143.9700	165.1839	212.1624	281.9558	374.6952	490.3750	628.9953	790.5561	975.0575	1182.4993	1412.8817	1666.2047	1942.4681	2241.6721
MAINTAINA	26.0000	40.6301	52.1237	71.9970	99.7375	135.3680	178.8876	230.2963	289.5941	356.7810	431.8570	514.8221	605.6763	704.4195	811.0519
PERIODICI	16.0000	18.6571	23.9714	31.9429	42.5714	55.8571	71.8000	90.4000	111.6571	135.5714	162.1429	191.3714	223.2571	257.8000	295.0000
RELIABILIT	4.0000	5.9493	7.5856	10.3754	14.2533	19.2223	25.2822	32.4331	40.6749	50.0077	60.4315	71.9461	84.5518	98.2483	113.0359
SIZE	184.2100	184.4839	185.0316	185.8533	186.9488	188.3182	189.9615	191.8787	194.0698	196.5347	199.2735	202.2862	205.5728	209.1333	212.9676

Figura 4-20 – Dados gerados pela simulação.

A Tabela 4-15 apresenta uma comparação entre as tendências do comportamento esperado para o sistema e as tendências do comportamento obtidas através da simulação, utilizando o novo conjunto de dados do sistema em observação.

Tabela 4-15 – Comparação entre comportamento Esperado e Simulado.

Comportamento	Tamanho	Periodicidade	Complexidade	Esforço	Confiabilidade	Manutenibilidade
Esperado	↑	↑	↑	↑	↓	↓
Simulado	↑	↑	↑	↑	↓	↓

O comportamento esperado para as características de software representa a situação real do sistema em um determinado período do tempo, considerando que o comportamento do sistema se mantenha nas versões seguintes (ARAÚJO 2009).

Com os resultados da simulação pode-se observar que o comportamento esperado, para todas as características analisadas, corresponde ao comportamento

simulado. Para verificar a coerência dos resultados da simulação é necessário coletar dados de versões futuras do sistema.

4.4.3 Situação Real do Sistema x Resultados da Simulação (Segunda Coleta de Dados)

Para analisar os resultados da simulação apresentados na seção anterior, serão apresentados os dados coletados de versões posteriores às coletadas para a construção das equações de simulação. Assim, será possível verificar a situação real do sistema e comparar com os resultados da simulação.

A Tabela 4-16 apresenta os dados coletados de 5 novas versões do sistema em observação, posteriores aos coletados para a construção das equações de simulação. As métricas são as mesmas utilizadas anteriormente.

Tabela 4-16 – Dados coletados para cinco versões do sistema em observação.

#	Tamanho	Periodicidade	Complexidade	Esforço	Confiabilidade	Manutenibilidade
1	186,67	3,00	36,39	54,68	3,00	16,00
2	187,81	6,00	36,56	65,97	3,00	19,00
3	189,32	7,00	36,83	79,62	9,00	37,00
4	191,06	6,00	37,27	76,03	14,00	14,00
5	198,26	18,00	38,74	151,17	8,00	30,00

Após a análise individual de cada versão sobre comportamento das características de software foram apresentadas as tendências para cada uma delas, assim como apresentado na Tabela 4-17. O comportamento individual das características é apresentado como comportamento observado, representando a situação real do sistema, correspondente ao período analisado.

Com os resultados da situação real do sistema, tanto de um momento anterior quanto de um momento posterior ao procedimento de simulação, foi possível avaliar os resultados obtidos pela simulação. A Tabela 4-17 apresenta um comparativo entre o comportamento esperado, baseado nas versões utilizadas para construção das equações do modelo, o comportamento simulado e o comportamento observado, baseado em novas versões do sistema.

Tabela 4-17 – Comparação entre comportamento Esperado, Simulado e Observado.

Comportamento	Tamanho	Periodicidade	Complexidade	Esforço	Confiabilidade	Manutenibilidade
Esperado	↑	↑	↑	↑	↓	↓
Simulado	↑	↑	↑	↑	↓	↓
Observado	↑	↑	↑	↑	↓	↓

Como se pode notar, os comportamentos esperados e observados do sistema, para todas as características de software analisadas, correspondem exatamente aos comportamentos simulados.

De acordo com ARAÚJO (2009), a Tabela 4-18 apresenta a interpretação do estado de cada Lei de Evolução de Software de acordo com os resultados da simulação.

Tabela 4-18 – Interpretação do estado das Leis de Evolução de Software (ARAÚJO 2009).

Lei	Implicação	Interpretação
Mudança Contínua	Negativa	Como a periodicidade foi medida através do intervalo de tempo em dias entre as versões e como essa medida apresenta significativo aumento, isso pode indicar que novas versões tendem a demorar mais para serem liberadas, caracterizando que a mudança contínua pode estar comprometida.
Incremento da Complexidade	Positiva	Isso pode acontecer porque o tamanho e a complexidade estão aumentando, o que é considerado como implicação positiva, podendo evidenciar que o sistema está em evolução.
Autorregulação	Negativa	As características Tamanho e Confiabilidade estão se comportando de maneira oposta ao esperado para a implicação positiva, o que pode evidenciar uma falta de controle no processo de desenvolvimento e manutenção do sistema.
Conservação da Estabilidade Organizacional	Negativa	Pode ocorrer em função do elevado aumento do esforço durante o processo de simulação, podendo evidenciar que não existe uma estabilidade por parte da equipe participante do projeto.
Conservação da Familiaridade	Negativa	O aumento do tamanho, da complexidade e do esforço pode indicar que o sistema será bastante modificado. Um sistema sendo modificado pode não estar mantendo a sua familiaridade.
Crescimento Contínuo	Negativa	Pode ocorrer em função de a periodicidade estar aumentando, o que pode caracterizar que novas versões do sistema tendem a demorar cada vez mais a surgirem, o que, por sua vez, pode retratar uma estagnação em suas funcionalidades.
Declínio da Qualidade	Positiva	O esforço, a complexidade e a manutenibilidade estão aumentando, o que pode indicar uma perda de qualidade do sistema no futuro.

Novamente, é importante ressaltar que esses resultados são específicos para o conjunto de dados utilizado, e que, portanto, os resultados não podem ser generalizados.

4.5 Análise do Impacto dos Novos Relacionamentos no Modelo

Como foi possível observar, a inclusão dos dois novos relacionamentos no modelo de Dinâmica de Sistemas causaram um impacto positivo nos resultados obtidos com a simulação. Entretanto, a partir disso, surge um questionamento sobre qual dos dois relacionamentos causou maior impacto nos resultados.

Para analisar essa questão, dois novos estudos de simulação foram realizados. Para evitar a confusão com estudos anteriores, os dois novos estudos foram identificados como Estudo A e Estudo B. Esses estudos foram executados somente com os dados da primeira coleta para possibilitar a comparação com os resultados do modelo original obtidos por ARAÚJO (2009).

4.5.1 Estudo A

Consistiu na inclusão apenas do relacionamento entre as características de software Tamanho e Manutenibilidade no modelo original de Dinâmica de Sistemas para observar as alterações causadas por esse relacionamento no modelo. Portanto, o relacionamento entre as características de software Esforço e Manutenibilidade não foi considerado.

A Tabela 4-19 apresenta uma comparação entre os comportamentos observados (comportamento real do sistema) e os resultados dos comportamentos obtidos pela simulação do modelo original e do modelo do estudo A (modelo original + relacionamento entre Tamanho e Manutenibilidade) e também os resultados obtidos com o novo modelo de observação de evolução de software, que considera tanto o relacionamento entre Tamanho e Manutenibilidade quanto o relacionamento entre Esforço e Manutenibilidade.

Tabela 4-19 – Comparação entre os comportamentos para análise do impacto da inclusão do relacionamento entre Tamanho e Manutenibilidade.

Comportamento	Tamanho	Periodicidade	Complexidade	Esforço	Confiabilidade	Manutenibilidade
Observado	↑	↑	↑	↑	↓	↑
Simulado Modelo Original	↑	↑	↑	↑	↓	↔
Simulado Modelo do Estudo A	↑	↑	↑	↑	↓	↔
Simulado Novo Modelo	↑	↑	↑	↑	↓	↓

Como se pode observar, a inclusão apenas do relacionamento entre Tamanho e Manutenibilidade (representado pelo Modelo do Estudo A) não causou um impacto significativo nos resultados da simulação, já que os resultados obtidos com o modelo original não foram alterados.

4.5.2 Estudo B

Consistiu na inclusão apenas do relacionamento entre as características de software Esforço e Manutenibilidade no modelo original, ou seja, dessa vez o relacionamento entre Tamanho e Manutenibilidade não foi considerado na simulação.

A Tabela 4-20 apresenta uma comparação entre os resultados dos comportamentos obtidos com a simulação do modelo original e do modelo do estudo B (modelo original + relacionamento entre Esforço e Manutenibilidade) e também os resultados obtidos com o novo modelo de observação de evolução de software.

Tabela 4-20 – Comparação entre os comportamentos para análise do impacto da inclusão do relacionamento entre Tamanho e Manutenibilidade.

Comportamento	Tamanho	Periodicidade	Complexidade	Esforço	Confiabilidade	Manutenibilidade
Observado	↑	↑	↑	↑	↓	↑
Simulado Modelo Original	↑	↑	↑	↑	↓	↔
Simulado Modelo Estudo B	↑	↑	↑	↑	↓	↓
Simulado Novo Modelo	↑	↑	↑	↑	↓	↓

Como se pode observar a inclusão do relacionamento entre as características Esforço e Manutenibilidade causou um impacto significativo nos resultados da simulação, já que o comportamento da característica Manutenibilidade foi alterado.

Novamente, é importante ressaltar que o relacionamento entre as características Esforço e Manutenibilidade forma um *loop* entre as características Esforço, Manutenibilidade e Confiabilidade, e por esse motivo sua inclusão causa um impacto significativo nos resultados da simulação.

4.5.3 Conclusões Parciais

Com os resultados dos dois novos estudos de simulação, Estudo A e Estudo B, pode-se observar que a alteração no modelo original de Dinâmica de Sistemas que efetivamente alterou os resultados do comportamento das características foi a inclusão do relacionamento entre as características Esforço e Manutenibilidade.

Esse fato nos faz questionar a relevância de cada relacionamento no modelo de observação de evolução e abre portas para novos estudos visando identificar os relacionamentos mais importantes no modelo.

É importante ressaltar que esses resultados são específicos para os conjuntos de dados utilizados, portanto, para se ter um maior poder de generalização é preciso executar estudos de simulação com dados de outros projetos.

4.6 *Illium Software Evolution* x Vensim PLE

Esta seção apresenta os resultados de duas simulações executadas utilizando o novo modelo de observação de evolução de software apresentado neste trabalho. A primeira simulação foi realizada utilizando a ferramenta *Illium Software Evolution* (ARAÚJO 2009) e a segunda simulação, utilizando a ferramenta Vensim PLE. O objetivo das simulações é possibilitar uma comparação entre resultados obtidos através das duas ferramentas. Essa comparação visa garantir a consistência entre os modelos de Dinâmica de Sistemas construídos nas duas ferramentas e realizar uma avaliação dos resultados apresentados pela ferramenta *Illium Software Evolution* frente a uma ferramenta comercial, amplamente utilizada tanto na academia quanto na indústria. Dessa forma, pode-se conferir uma maior confiança aos resultados dos estudos de simulação apresentados neste trabalho.

Apesar da análise dos resultados ser semiquantitativa, a Tabela 4-21 apresenta os valores dos resultados das simulações executadas nas ferramentas *Illium Software Evolution* e Vensim PLE, ambas utilizando o novo modelo de observação de evolução de software. O objetivo dessa tabela é possibilitar uma comparação detalhada entre os resultados.

Tabela 4-21 – Resultados da simulação do novo modelo em Vensim PLE (V) e *Illium Software Evolution (I).*

Tempo	Tamanho		Periodicidade		Complexidade		Esforço		Confiabilidade		Manutenibilidade	
	V	I	V	I	V	I	V	I	V	I	V	I
0	63,29	63,29	4,00	4,00	11,43	11,43	0,50	0,50	1,00	1,00	0,50	0,50
1	63,29	63,29	4,00	4,00	11,43	11,43	1,50	1,50	1,17	1,17	1,51	1,51
2	63,30	63,30	4,23	4,23	11,43	11,43	2,27	2,27	1,34	1,34	2,25	2,25
3	63,31	63,31	4,68	4,68	11,43	11,43	2,99	2,99	1,53	1,53	2,91	2,91
4	63,33	63,33	5,36	5,36	11,43	11,43	3,81	3,81	1,78	1,78	3,63	3,63
5	63,35	63,35	6,28	6,28	11,44	11,44	4,83	4,83	2,09	2,09	4,51	4,51
6	63,38	63,38	7,41	7,41	11,44	11,44	6,14	6,14	2,49	2,49	5,64	5,64
7	63,42	63,42	8,77	8,77	11,45	11,45	7,79	7,79	2,99	2,99	7,09	7,09
8	63,46	63,46	10,36	10,36	11,46	11,46	9,84	9,84	3,59	3,59	8,91	8,91
9	63,51	63,51	12,18	12,18	11,46	11,46	12,33	12,33	4,29	4,29	11,13	11,13
10	63,56	63,56	14,23	14,23	11,47	11,47	15,29	15,29	5,12	5,12	13,78	13,78
11	63,62	63,62	16,50	16,50	11,48	11,48	18,74	18,74	6,06	6,06	16,89	16,89
12	63,68	63,68	19,00	19,00	11,49	11,49	22,69	22,69	7,13	7,13	20,48	20,48
13	63,75	63,75	21,73	21,73	11,51	11,51	27,17	27,17	8,32	8,32	24,56	24,56
14	63,83	63,83	24,68	24,68	11,52	11,52	32,17	32,17	9,63	9,63	29,14	29,14
15	63,91	63,91	27,86	27,86	11,53	11,53	37,72	37,72	11,08	11,08	34,22	34,22
16	63,99	63,99	31,27	31,27	11,55	11,55	43,81	43,81	12,65	12,65	39,82	39,82
17	64,09	64,09	34,91	34,91	11,56	11,56	50,45	50,45	14,35	14,35	45,94	45,94
18	64,19	64,19	38,77	38,77	11,58	11,58	57,64	57,64	16,18	16,18	52,58	52,58
19	64,29	64,29	42,86	42,86	11,59	11,59	65,39	65,39	18,14	18,14	59,75	59,75
20	64,40	64,40	47,18	47,18	11,62	11,62	73,70	73,70	20,23	20,23	67,45	67,45

Os resultados apresentados na Tabela 4-21 permitem concluir que a ferramenta *Illium Software Evolution* fornece exatamente os mesmos resultados da ferramenta Vensim PLE para um mesmo modelo de dinâmica de sistemas e um mesmo conjunto de dados. Dessa forma, pode-se garantir maior confiança nos resultados apresentados neste trabalho.

Com base nos dados apresentados na Tabela 4-21, as tendências de cada característica de software analisada foram calculadas e apresentadas na Tabela 4-22.

Tabela 4-22 – Comparação dos resultados da tendência de crescimento entre *Illium* e Vensim para o modelo original

Comportamento	Tamanho	Periodicidade	Complexidade	Esforço	Confiabilidade	Manutenibilidade
Simulado *Illium*	↑	↑	↑	↑	↓	↓
Simulado Vensim	↑	↑	↑	↑	↓	↓

Pode-se observar que os resultados da simulação obtidos com a utilização do simulador Vensim PLE foram os mesmos resultados obtidos com o simulador *Illium Software Evolution*. Portanto, podemos concluir que o *Illium Software Evolution* está coerente na implementação da simulação utilizando a técnica de Dinâmica de Sistemas.

4.7 Conclusão

Este capítulo apresentou as alterações realizadas no modelo de Dinâmica de Sistemas, os resultados das simulações com o novo modelo e uma comparação com os resultados da simulação do modelo original.

Com base nos resultados das simulações executadas com o modelo original e com o novo modelo foi possível observar que as alterações realizadas no modelo de Dinâmica de Sistemas relacionadas à inclusão dos novos relacionamentos alteraram significativamente os comportamentos simulados.

Tendo em vista que os comportamentos simulados obtidos com a primeira coleta de dados não corresponderam aos comportamentos observados naquele momento, em que o foco do desenvolvimento estava concentrado na implementação de novas funcionalidades, e que, por outro lado, os comportamentos simulados obtidos com a segunda coleta de dados corresponderam aos comportamentos observados, e que nesse momento o foco do desenvolvimento estava concentrado em atividades de testes beta e manutenção, é possível concluir que o contexto do momento da coleta de dados é importante para a interpretação dos resultados.

A análise apresentada sobre o impacto da inclusão dos dois novos relacionamentos no modelo permite afirmar que o relacionamento entre as características Esforço e Manutenibilidade foi o responsável pela alteração nos resultados da simulação.

Também foram apresentadas modificações no modelo de Dinâmica de Sistemas que permitem que novos comportamentos sejam observados durante os estudos de simulação. Além disso, foi apresentada uma comparação entre os resultados da simulação das duas ferramentas utilizadas nesse trabalho para dar maior validade aos resultados apresentados.

No próximo capítulo serão apresentadas diretrizes sobre a execução do procedimento de simulação e também uma ferramenta Web desenvolvida para apoiar a execução do procedimento.

5 Diretrizes para Utilização do Modelo de Evolução de Software

Para facilitar e difundir a utilização do modelo de observação de evolução de software apresentado neste trabalho, nesse capítulo será apresentada uma visão geral do procedimento de simulação e um conjunto de diretrizes definidos para apoiar sua execução. Além disso, também será apresentada uma ferramenta Web desenvolvida com o propósito de apoiar procedimento de simulação através da automatização dos passos do procedimento. A ferramenta foi chamada de WISE – *Web Illium Software Evolution,* por ser baseada na ferramenta *Illium Software Evolution* (ARAÚJO 2009) e ter a Web como plataforma nativa.

As diretrizes e a ferramenta são extremamente importantes para facilitar a execução de futuros estudos na área de evolução de software que tenham interesse em utilizar, avaliar, testar ou mesmo conhecer o modelo de observação evolução de software apresentado nesse trabalho.

As diretrizes apresentadas foram definidas com base nas experiências obtidas pelos pesquisadores durante a execução dos estudos de simulação e contêm informações sobre a coleta dos dados necessários para a execução das simulações e informações sobre a utilização da ferramenta WISE.

5.1 O Procedimento de Simulação

Para facilitar o entendimento do procedimento de simulação utilizado para executar estudos de simulação baseados no novo modelo de observação de evolução de software foi elaborado um fluxograma, apresentado na Figura 5-1, contendo os passos necessários para a execução do procedimento.

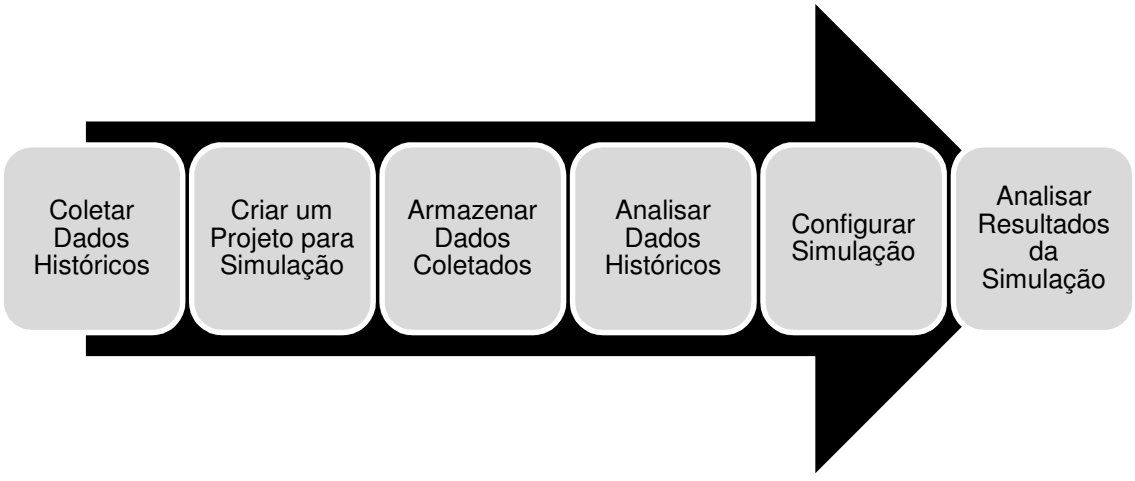


Figura 5-1 – Passos do procedimento de simulação.

O primeiro passo do procedimento de simulação é a coleta de dados históricos do sistema que se deseja observar.

O segundo passo consiste na criação de um projeto para simulação. Nesse momento serão registradas informações sobre o projeto, a fase do desenvolvimento que será analisada e as métricas utilizadas.

O terceiro passo consiste no armazenamento dos dados históricos coletados. Os dados são inseridos de acordo com cada versão analisada do sistema em observação.

O quarto passo consiste na análise dos dados armazenados. São sugeridos dois tipos de análise. O primeiro é uma análise visual, através de gráficos de tendências, para os dados históricos de cada uma das características para possibilitar uma compreensão geral da situação da evolução do sistema. O segundo é uma análise realizada através do desvio padrão, calculado de forma cumulativa entre as versões. Essa análise permite avaliar o grau de variação do conjunto de dados utilizado para auxiliar na definição do menor número de versões que precisam ser utilizadas na calibração do modelo. De acordo com ARAÚJO (2009), a quantidade de versões que apresentam o menor desvio padrão pode ser considerada como o conjunto mínimo para iniciar o procedimento de simulação. Nesse momento o usuário define o conjunto mínimo de versões que serão utilizadas na simulação.

O quinto passo consiste na configuração da simulação. Nessa etapa o usuário define a quantidade de passos de simulação que serão executados e também o valor inicial de cada característica de software analisada.

O sexto passo consiste na análise dos resultados da simulação. Essa análise pode ser feita com base nos resultados da simulação e em gráficos de tendências calculados a partir desses dados. Outra opção é fazer uma comparação entre os dados antes e depois da simulação, analisando o comportamento esperado e o comportamento simulado com base nas tendências.

5.2 Diretrizes para Coleta de Dados

Para apoiar o procedimento de coleta de dados foram definidas diretrizes descrevendo detalhes sobre como os dados devem ser utilizados para executar a simulação.

1) O modelo considera que a observação da evolução de um produto de software é feita em diferentes etapas do processo de desenvolvimento (especificação de requisitos, projeto de alto nível, projeto de baixo nível e

codificação) e, para isso, não se devem misturar métricas de diferentes etapas para a observação;

2) Apenas uma métrica daquelas listadas na Tabela 2-7 é necessária para representar cada uma das características e apenas medidas para essas métricas devem ser coletadas para a observação da evolução do produto de software;
3) As características de software estão intimamente relacionadas à capacidade de observação da evolução do software. A falta de medidas para a métrica de uma determinada característica impossibilita a visualização de seu comportamento e influencia na observação do comportamento das demais características presentes no modelo;
4) No sentido de viabilizar a observação da evolução do software, é importante que o mesmo possua repositórios com dados históricos relacionados às métricas escolhidas para representar as características. Caso contrário, o processo de simulação não será possível;
5) Para facilitar a coleta de dados é importante que durante o ciclo de desenvolvimento e manutenção do software todos os artefatos do produto relacionados às etapas do desenvolvimento em observação sejam congelados para cada versão do software;
6) A medição deve ser realizada para um conjunto, com pelo menos, 5 versões do software em observação de forma a oferecer um mínimo de estabilidade para o procedimento de simulação. Caso existam mais versões disponíveis, sugere-se que sejam também utilizadas visando a reduzir o erro estatístico na simulação;
7) As medições das diferentes versões do software em observação devem ser realizadas sempre para as mesmas métricas e nas mesmas unidades;
8) A qualidade das medições realizadas deve ser garantida previamente a partir da análise da variabilidade das medidas e possíveis discrepâncias observadas;
9) É importante que sejam coletadas medidas de versões sucessivas do software em observação, excetuando quando alguma versão é desconsiderada por motivo de falta de qualidade nos dados. Nessa situação, deve-se ter atenção ao avaliar o comportamento das características de software, uma vez que sofreram uma descontinuidade na observação;
10) Os dados históricos referentes às diferentes versões devem ser utilizados obedecendo a ordem cronológica, conforme registrado nos repositórios;

11) O procedimento de simulação resulta na indicação de tendências do comportamento das características de software em função de seus dados históricos. Desta forma, é importante verificar se o comportamento apresentado pela simulação está coerente com o comportamento das versões seguintes do software em observação, desconsiderando o conjunto de dados utilizado para a simulação.

5.3 WISE – Web Illium Software Evolution

A ferramenta WISE foi desenvolvida com o propósito principal de apoiar a execução do procedimento de simulação para facilitar a utilização do novo modelo de observação de evolução de software apresentado nesse trabalho.

O fácil acesso ao modelo aumenta a possibilidade de novos estudos serem realizados. Desta forma, novas contribuições para a área de evolução de software podem ser apresentadas. Para isso, a ferramenta foi desenvolvida em uma plataforma Web e, além disso, foram acrescentadas novas funcionalidades para facilitar a execução do procedimento de simulação apresentado e a interpretação dos resultados obtidos.

Assim como a ferramenta *Illium Software Evolution,* desenvolvida por ARAÚJO (2009), a ferramenta WISE é uma interface que utiliza a máquina de simulação de *Illium*, desenvolvida por BARROS (2000). Portanto, os resultados apresentados pela nova ferramenta são os mesmos apresentados por *Illium Software Evolution* (ARAÚJO 2009). Entretanto, foram adicionadas novas formas de visualização dos resultados e novas funcionalidades para automatizar o procedimento de simulação.

5.3.1 Principais Características

A ferramenta WISE automatiza praticamente todos os passos do procedimento de simulação descrito anteriormente. Apenas o passo da coleta dados não é totalmente apoiado. Como descrito anteriormente, *Illium Software Evolution* apóia apenas os passos de configuração da simulação e análise dos resultados, todos os outros passos eram realizados manualmente.

As características que diferenciam a ferramenta WISE da ferramenta *Illium Software Evolution* são descritas a seguir.

- Adição de uma funcionalidade que permite a criação de um projeto com informações sobre a etapa do desenvolvimento de software a ser analisada e informações sobre as métricas a serem utilizadas;

- Adição de uma funcionalidade que permite o armazenamento de dados históricos de um projeto, permitindo que as simulações sejam reexecutadas a qualquer momento e, além disso, permite que o observação da evolução de um projeto seja feita a longo prazo;
- Adição de uma funcionalidade que permite gerar gráficos automaticamente a partir dos dados históricos cadastrados para cada característica. Com isso é possível observar o comportamento de tendência de cada característica de forma individual antes e depois da simulação;
- Adição de uma funcionalidade que permite a análise estatística dos dados, através da análise do desvio padrão cumulativo entre as versões cadastradas do sistema, para cada característica de software;
- Automatização da criação do modelo de Dinâmica de Sistemas, através do cálculo automático dos coeficientes das equações do modelo. Em *Illium Software Evolution,* o cálculo era feito através de uma planilha Excel;
- Adição de uma funcionalidade que permite a alteração dos dados utilizados como entrada para o modelo de simulação, ou seja, os valores iniciais de cada característica. Dessa forma é possível interagir com o modelo de simulação. Em *Illium Software Evolution*, os dados iniciais de cada característica eram obtidos estaticamente a partir dos valores da última versão do sistema analisado;
- Funcionalidade que permite comparar os resultados da simulação com o comportamento esperado e observado do sistema em observação.

5.4 Diretrizes para Execução do Procedimento de Simulação

A apresentação da ferramenta WISE será feita com os novos dados coletados do sistema em observação apresentados na Tabela 4-12. Assim, será possível mostrar que os resultados apresentados pela ferramenta WISE são os mesmos apresentados por *Illium Software Evolution*.

- **Criação de uma Conta**

A primeira etapa para a utilização da ferramenta é a criação de uma conta no sistema. Para isso o usuário precisa fornecer um nome de usuário e uma senha. Após realizar o cadastro, o usuário está apto para fazer login no sistema e seguir nas etapas do procedimento de simulação. A Figura 5-2 apresenta a tela para cadastro de usuário.

Figura 5-2 – Tela de Cadastro.

Como dito anteriormente, a ferramenta WISE fornece apoio a praticamente todo o procedimento de simulação apresentado na Figura 5-1. A seguir, serão apresentadas as etapas do procedimento de simulação apoiadas pela ferramenta.

1º Passo – Definir Projeto

O primeiro passo do procedimento de simulação suportado pela ferramenta é a definição de um projeto de simulação para armazenar os dados históricos coletados de um sistema. No primeiro passo o usuário tem a possibilidade de criar um projeto de simulação ou selecionar algum projeto criado anteriormente.

- **Criação do Projeto**

Na criação do projeto o usuário deve fornecer as seguintes informações:

Nome – Nome de identificação do projeto.

Etapa – Etapa do processo de desenvolvimento a ser observada.

Descrição – Informações sobre o projeto, sobre o processo de coleta de dados realizado e o contexto da situação do desenvolvimento no momento da coleta.

Métrica para Tamanho – Informar a métrica utilizada para medir o Tamanho.

Métrica para Periodicidade – Informar a métrica utilizada para medir a Periodicidade.

Métrica para Complexidade – Informar a métrica utilizada para medir a Complexidade.

Métrica para Esforço – Informar a métrica utilizada para medir o Esforço.

Métrica para Confiabilidade – Informar a métrica utilizada para medir a Confiabilidade.

Métrica para Manutenibilidade – Informar a métrica utilizada para medir a Manutenibilidade.

Após salvar o projeto o usuário é encaminhado para o segundo passo do procedimento de simulação. A Figura 5-3 apresenta a tela de criação do projeto.

Figura 5-3 – Tela de Criação de Projeto.

- **Selecionar Projeto**

Ainda no primeiro passo, o usuário pode selecionar um projeto de simulação criado anteriormente. Para isso o usuário deve clicar em selecionar projetos. E para escolher um dos projetos listados o usuário deve clicar no ícone . Após selecionar um projeto, o usuário é encaminhado para o segundo passo do procedimento de simulação. A Figura 5-4 apresenta a tela de listagem dos projetos cadastrados no sistema.

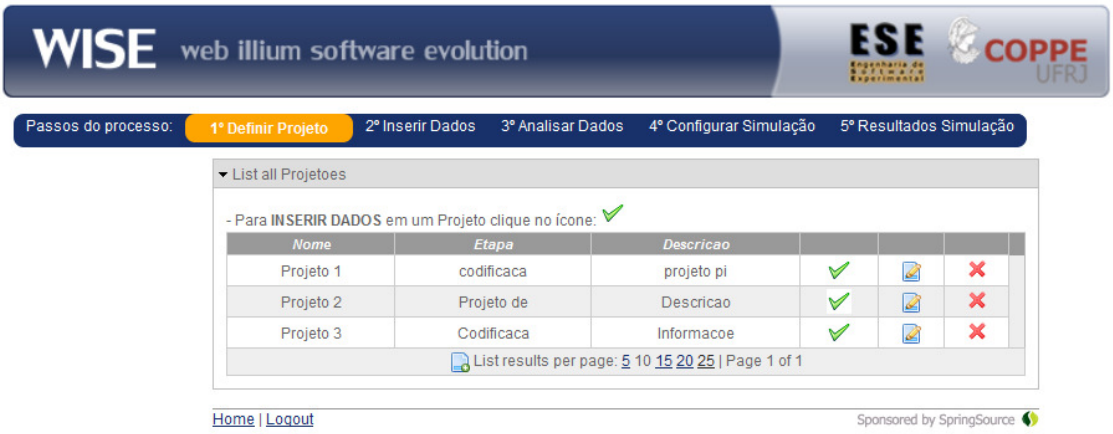


Figura 5-4 – Tela de seleção de projetos.

2º Passo – Inserir Dados Históricos

O segundo passo do procedimento de simulação apoiado pela ferramenta é a inserção dos dados históricos coletados. Os dados devem ser inseridos separadamente por versão analisada do sistema. Além do formulário para inserção de dados, a tela apresenta as informações sobre o projeto, as métricas utilizadas e os dados históricos já armazenados para o projeto. De acordo com as diretrizes para coleta de dados apresentadas anteriormente, devem ser inseridos dados de pelo menos cinco versões do sistema. A Figura 5-5 apresenta a tela de inserção de dados.

Para seguir para o terceiro passo, o usuário deve clicar no botão “Analisar Dados”, localizado no final ta tela.

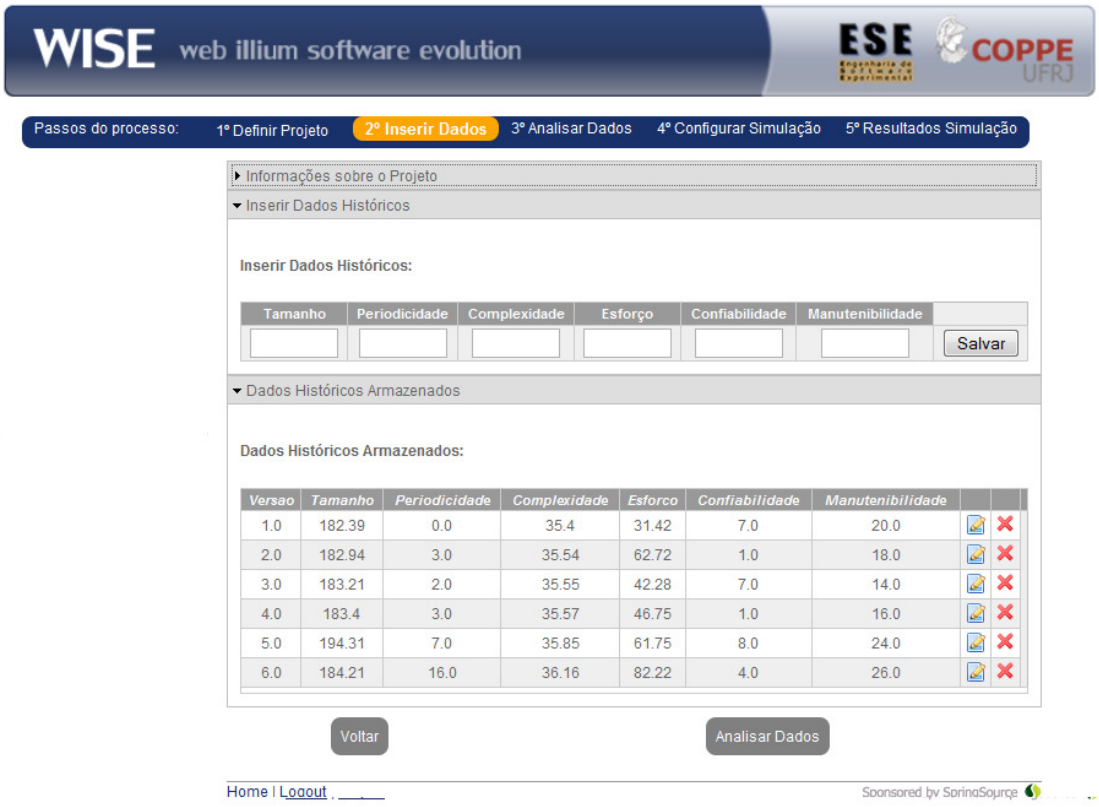


Figura 5-5 – Tela de inserção de dados históricos.

3º Passo – Análise dos Dados Históricos

No terceiro passo do procedimento de simulação apoiado pela ferramenta são apresentadas as informações sobre o projeto, os dados armazenados e três opções para analisar os dados. A primeira opção é uma análise do desvio padrão calculado de forma cumulativa entre as versões. A segunda são os gráficos das tendências de cada característica e a terceira o comportamento esperado, ambos calculados a partir dos dados históricos armazenados. Para visualizar os gráficos o usuário precisa clicar no

ícone  . Nesse momento, o usuário tem a possibilidade de desconsiderar alguma versão do sistema que contenha dados considerados de baixa qualidade, com base na análise dos dados, clicando no botão  .

Para seguir para o quarto passo o usuário deve clicar no botão “Configurar Simulação”, localizado no final ta tela. As figuras 5-6 e 5-7 apresentam a tela de analise dos dados.

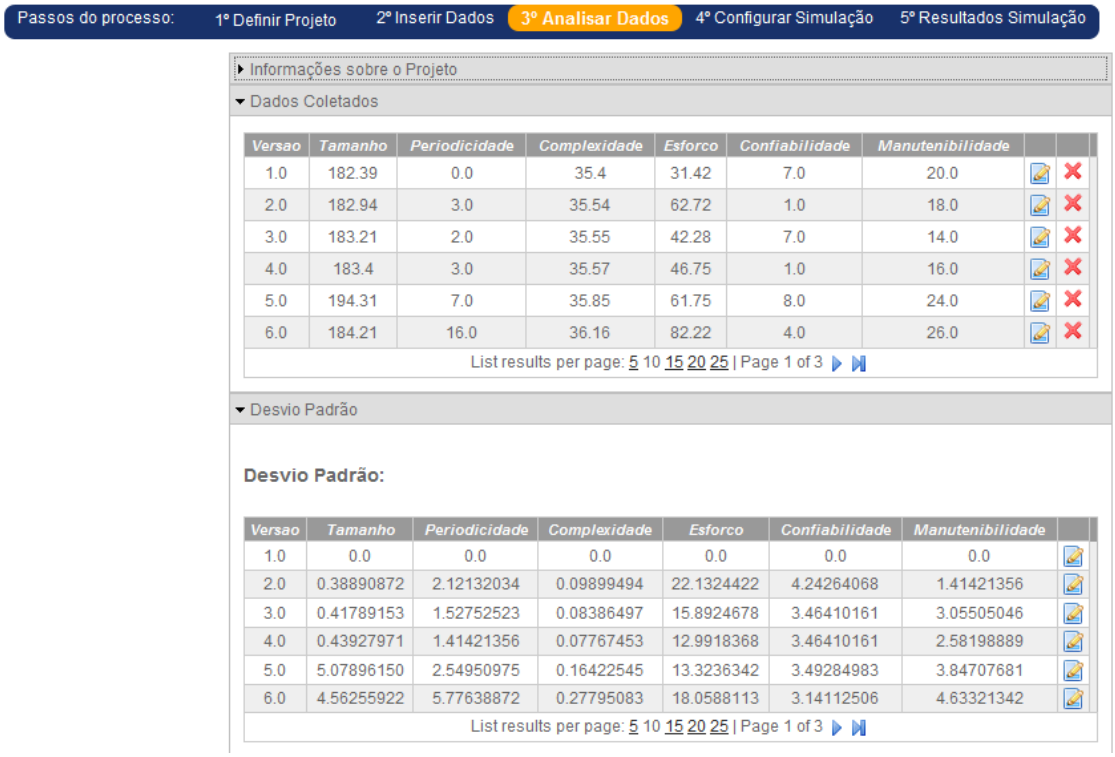


Figura 5-6 – Primeira parte da tela de análise dos dados históricos.

Figura 5-7 – Segunda parte da tela de análise dos dados históricos.

4º Passo – Configuração da Simulação

O quarto passo do procedimento de simulação apoiado pela ferramenta consiste na configuração do simulador. Nesse momento o usuário define o número de passos de simulação a serem executados e também os valores iniciais para cada variável do modelo que corresponde às características de software analisadas. Inicialmente, o valor de cada característica vem preenchido com os valores da última versão do sistema armazenada no segundo passo. Além disso, também são mostradas as informações sobre o projeto de simulação. Para seguir para o quinto passo o usuário deve clicar no botão "Executar Simulação", localizado no final ta tela. Ao clicar nesse botão a ferramenta irá passar o modelo de Dinâmica de Sistemas para a máquina de simulação de *Illium* (2000) que irá realizar a simulação de fato. A Figura 5-8 apresenta a tela de configuração da simulação.

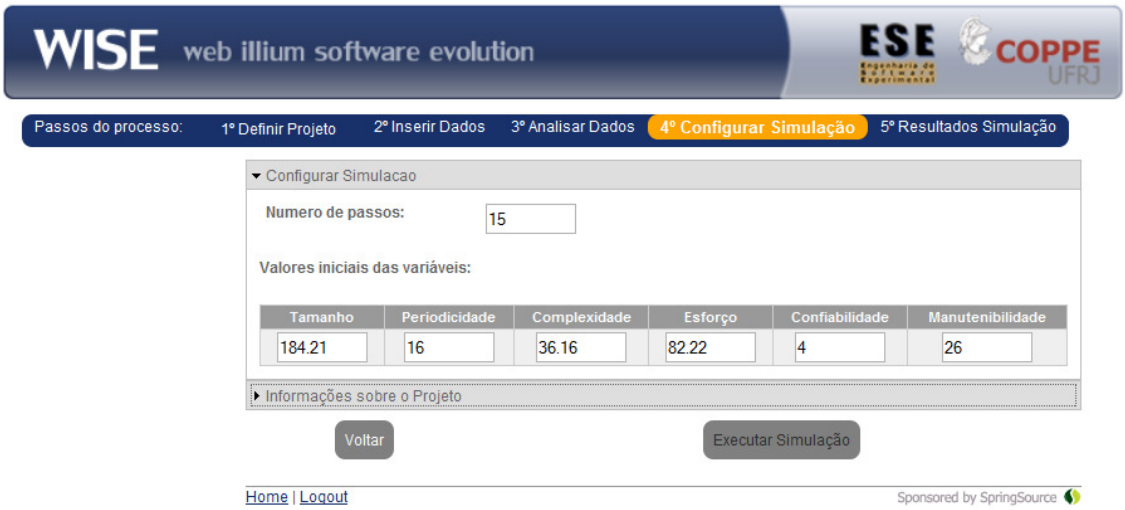


Figura 5-8 – Tela de configuração da simulação.

5º Passo – Resultados da Simulação

O quinto e último passo do procedimento de simulação apoiado pela ferramenta consiste na apresentação dos resultados da simulação. Nesse momento o usuário pode ver o modelo de Dinâmica de Sistemas, tanto na forma textual, apresentado na linguagem definida por *Illium* (2000), quanto no formato gráfico, apresentado nos padrões de modelagem da ferramenta Vensim PLE. O modelo de Dinâmica de Sistemas apresentado no formato gráfico corresponde ao modelo executado na simulação.

Os dados gerados pela simulação são apresentados em uma tabela contendo os passos de simulação, que correspondem a uma versão futura do sistema e os valores das características de software para cada versão.

Ao clicar no botão “Ver gráfico de todas as características” é apresentado um gráfico contendo a curva de todas as características analisadas, calculadas com base nos resultados da simulação. É importante lembrar que esse gráfico não possui escala no eixo y, já que cada característica foi medida com uma métrica diferente.

O gráfico da tendência de cada característica pode ser observado ao clicar no ícone  . Também é apresentada uma comparação entre o comportamento esperado, que é calculado com base nos dados históricos, e o comportamento simulado, que é calculado com base nos resultados da simulação.

Além disso, é apresentada a situação de cada Lei de Evolução de Software através de semáforos, onde a cor cinza significa implicação neutra, a cor verde implicação positiva e a cor vermelha implicação negativa. A situação de cada Lei é calculada com base na tendência de cada característica e nas premissas definidas por ARAÚJO (2009). As premissas podem ser vistas ao clicar no botão “Premissas de Observação das LES” e as interpretações sobre cada implicação poder ser observada ao clicar no botão “Interpretação das implicações”.

Nessa etapa o usuário tem a possibilidade de voltar ao passo anterior, onde as equações do modelo de simulação já foram calculadas, e interagir com o modelo alterando a configuração da simulação para observar diferentes comportamentos de acordo com a variação dos dados passados como valores iniciais para cada característica.

As figuras 5-9 e 5-10 apresentam a tela de resultados da simulação.

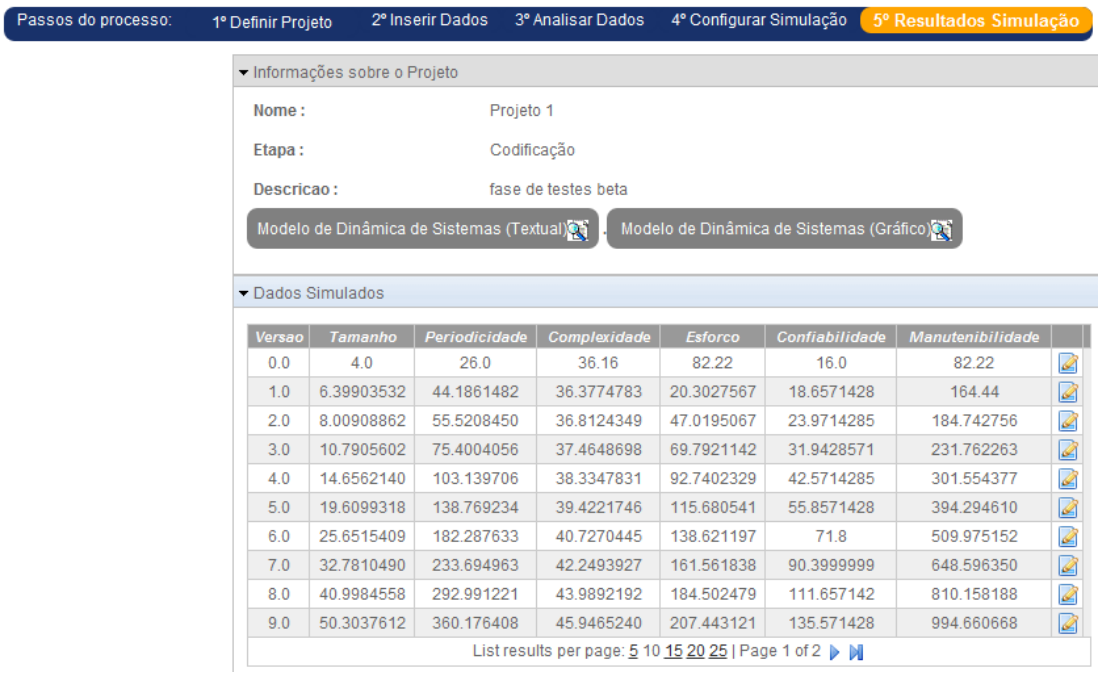
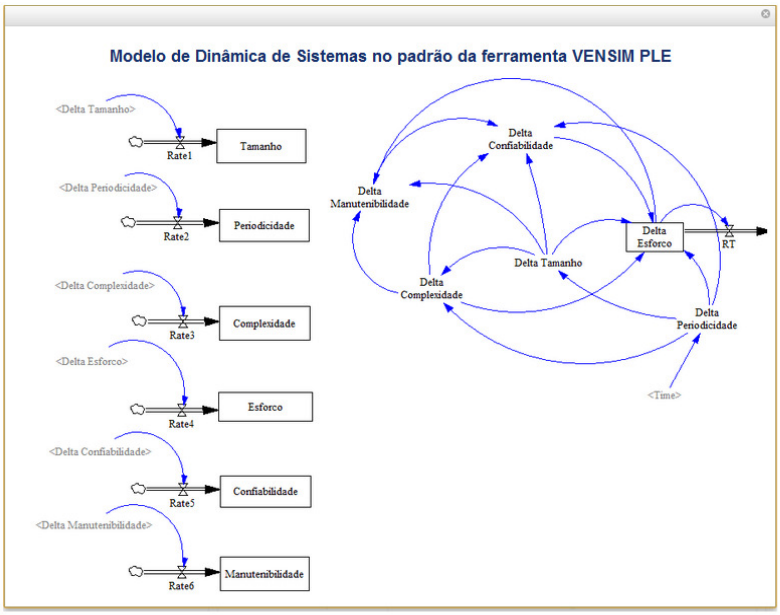


Figura 5-9 – Primeira parte da tela de resultados.

Figura 5-10 – Segunda parte da tela de resultados.

A Figura 5-11 apresenta a visualização do modelo tanto no formado textual, baseado na linguagem definida para *Illium,* quanto no formato gráfico, baseado nos padrões de modelagem da ferramenta Vensim PLE.



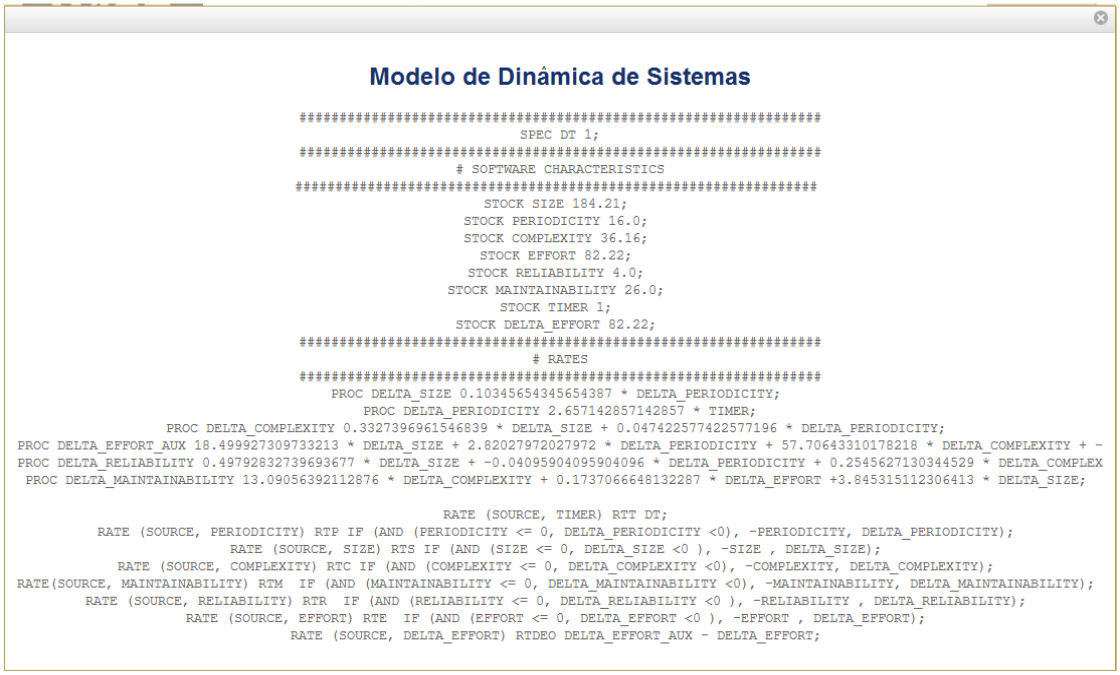


Figura 5-11 – Tela de apresentação dos modelos através da ferramenta WISE.

5.5 Conclusão

Nesse capítulo foram apresentados o procedimento de simulação, as diretrizes para a coleta de dados, as diretrizes para execução do procedimento de simulação e a ferramenta WISE, definida para apoiar o procedimento de simulação e a utilização do novo modelo de observação de evolução de software.

A definição de toda essa infraestrutura apresentada é importante para dar continuidade ao estudo apresentado neste trabalho, devido à maior facilidade em utilizar o modelo apresentado.

Apesar de a nova ferramenta tornar o processo de simulação automatizado e facilitar a execução de simulações, ela não foi avaliada experimentalmente para comprovar sua eficácia. O procedimento de simulação sugerido também não foi avaliado.

No próximo capítulo serão apresentadas as considerações finais, as contribuições deste trabalho, as limitações e os trabalhos futuros sugeridos.

6 Conclusões Finais

Nessa dissertação foram apresentados os resultados de uma avaliação conceitual de um modelo de observação de evolução de software. A avaliação ocorreu através da reexecução de um estudo baseado em revisão sistemática, que tinha como objetivo encontrar evidências adicionais na literatura técnica de Engenharia de Software sobre os relacionamentos apresentados no modelo original e verificar a validade conceitual dos relacionamentos apresentados.

Com os resultados do estudo foi possível atualizar o modelo e também incluir um novo relacionamento, originando assim um novo modelo de observação de evolução de software. Entretanto, não foram encontradas evidências adicionais para todos os relacionamentos apresentados no modelo, o que possibilita um questionamento sobre a relevância de cada um dos relacionamentos.

Após a atualização do modelo conceitual, o modelo de Dinâmica de Sistemas foi alterado de modo a refletir a alteração realizada. Durante essa etapa, foi identificado que um relacionamento apresentado pelo modelo conceitual estava ausente no modelo de simulação. Dessa forma, dois relacionamentos foram incluídos no modelo de Dinâmica de Sistemas. Além disso, foram realizadas alterações lógicas no cálculo das variações de cada característica, diminuindo restrições no cálculo, o que possibilita a observação de outros comportamentos através do novo modelo.

Para avaliar os resultados das alterações realizadas no modelo, foram executados estudos experimentais de simulação com dois conjuntos de dados de um mesmo sistema de software.

A comparação entre os resultados das simulações e a situação real do sistema permitiu afirmar que as alterações realizadas no modelo impactaram significativamente nos resultados.

6.1 Contribuições

A principal contribuição dessa pesquisa foi a atualização do modelo de observação de evolução de software.

Outras contribuições:

- Maior validade conceitual ao modelo de observação de evolução de software, devido à maior abrangência dada ao estudo baseado em revisão sistemática, reduzindo assim as ameaças à validade de construção do modelo;

- A descoberta de evidências adicionais para um novo relacionamento entre as características de software que resultou na inclusão de um novo relacionamento no modelo e conseqüentemente gerou um novo modelo de observação de evolução de software;
- Melhorias no modelo de Dinâmica de Sistemas através de modificações na lógica do cálculo realizado no modelo, permitindo que novos comportamentos sejam observados;
- Comparação entre os resultados obtidos com *Illium* (BARROS 2000) e Vensim PLE, dando maior validade aos resultados apresentados pela máquina de simulação *Illium* e também aos resultados apresentados pela ferramenta WISE, e;
- Ferramenta Web para automatizar o procedimento de simulação e facilitar a utilização da infraestrutura por outros grupos interessados na área de evolução de software.

6.2 Limitações

As principais limitações identificadas nessa pesquisa são:

- Não é possível confirmar a validade externa dos resultados apresentados nesse trabalho devido às simulações terem sido executadas com os dados de apenas um projeto;
- Os relacionamentos entre as características foram analisados separadamente por cada par de características e essa análise isolada por pares é um fator que pode enfraquecer a validade do modelo, e;
- O procedimento de simulação e a ferramenta desenvolvida para apoiar o procedimento não foram devidamente avaliados.

6.3 Trabalhos Futuros

Com o objetivo de dar continuidade ao estudo sobre evolução de software utilizando todo o conhecimento apresentado nesse trabalho, alguns trabalhos futuros são sugeridos a seguir:

- Executar estudos de simulação utilizando dados de outros projetos com o objetivo de verificar a validade do modelo apresentado para diferentes conjuntos de dados;
- Automatizar o processo de coleta de dados, integrando-o à ferramenta apresentada;

- Analisar os relacionamentos de forma mais integrada, ou seja, não avaliá-los somente de forma isolada por cada par de características;
- Análise da importância de cada relacionamento no modelo com o objetivo de encontrar um conjunto mínimo de relacionamentos para simplificar o modelo mantendo ou melhorando a qualidade dos resultados obtidos a partir das simulações;
- Realizar análises estatísticas mais rigorosas para avaliar a significância dos coeficientes dos relacionamentos;
- Apoiar o processo de tomada de decisão introduzindo na seção de resultados sugestões sobre medidas a serem tomadas de acordo com a análise do comportamento de um sistema de software, e;
- Avaliar experimentalmente o procedimento de simulação sugerido e a ferramenta apresentada.

REFERÊNCIAS BIBLIOGRÁFICAS

ABDEL-HAMID, T., MADNICK, S.E. 1991. **"Software Project Dynamics: An Integrated Approach"**, Englewood Cliffs, NJ: Prentice-Hall, Inc.

AGGARWAL, K., SINGH, Y., KAUR, A., MALHOTRA, R. 2006. "**Empirical Study of Object-Oriented Metrics",** in *Journal of Object Technology*, vol. 5. no. 8, November-December 2006, pp. 149-173.

ARAÚJO, M.A.P., TRAVASSOS, G.H., KITCHENHAM, B. (2005). **"Evolutive Maintenance: Observing Object-Oriented Software Decay",** Technical Report ES-672/05 -COPPE/UFRJ.

ARAÚJO, M.A.P. 2009. **"Um Modelo Para Observação de Evolução de Software",** Tese de Doutorado, COPPE/UFRJ – Universidade Federal do Rio de Janeiro.

BASILI, V.R., BRIAND, L., CONDON, S., KIM, Y.M., MELO, W.L., VALETT, J., 1996, **"Understanding and Predicting the Process of Software Maintenance Releases"**. In: Proc. 18th Int´l Conf. Software Engineering, Berlin, Germany, IEEE Press.

BARROS, M., WERNER, C., TRAVASSOS, G. 2000. **"Illium: Uma ferramenta de Simulação de Modelos Dinâmicos de Projetos de Software".** XIV Simpósio Brasileiro de Engenharia de Software, pp.355-358.

BELADY, L.A., LEHMAN. M.M., 1976, "A Model of Large Program Development". IBM Systems Jounal, vol. 15, no. 1.

BIOLCHINI, J., MIAN, P.G., NATALI, A.C., TRAVASSOS, G.H., 2005, **Systematic Review in Software Engineering: Relevance and Utility**. Technical Report. PESC - COPPE/UFRJ. url: http://www.cos.ufrj.br/uploadfiles/es67905.pdf

BOCCO, M., MOODY, D. and PIATTINI, M. 2005. **"Assessing the capability of internal metrics as early indicators of maintenance effort through experimentation".** *Research Articles. J. Softw. Maint. Evol.* 17, 3 (May 2005), 225-246. DOI=10.1002/smr.v17:3 http://dx.doi.org/10.1002/smr.v17:3.

CANFORA, G., GARCÍA, F., PIATTINI, M., RUIZ, F., VISAGGIO, C. A. 2005. **"A family of experiments to validate metrics for software process models".** *Journal of Systems and Software,* v.77 n.2, p.113-129, August 2005. DOI= http://dx.doi.org/10.1016/j.jss.2004.11.007

COOK, C.R., ROESCH, A., 1994, **"Real-Time Software Metrics".** Journal of Systems and Software, vol. 24, no. 3, Special issue of the best papers from the Oregon Workshop on Software Metrics, pp 223 – 237, ISSN 0164-1212

CRUZ-LEMUS, J., GENERO, M., and PIATTINI, M. 2007. **"Using Controlled Experiments for Validating UML Statechart Diagrams Measures".** In Software Process and Product Measurement. Lecture Notes In Computer Science, Vol. 4895. Springer-Verlag, Berlin, Heidelberg 129-138. DOI= http://dx.doi.org/10.1007/978-3-540-85553-8_11

DARCY, D., KEMERER, C., SLAUGHTER, S., and TOMAYKO, J. 2005. **"The Structural Complexity of Software: An Experimental Test"**. *IEEE Trans. Softw. Eng.* 31, 11 (November 2005), 982-995. DOI=http://dx.doi.org/10.1109/TSE.2005.130

DOLADO JJ. 2001. **"On the problem of the software cost function".** *Inform Software Technology* 2001;43:61-72. DOI= http://dx.doi.org/10.1016/j.advengsoft.2004.10.001

EL EMAM, K. , BENLARBI, S., GOEL, N., MELO, W., LOUNIS, H., RAI, S. N. 2002. **"The Optimal Class Size for Object-Oriented Software"**. *IEEE Transactions on Software Engineering*, v.28 n.5, p.494-509, May 2002. DOI= http://dx.doi.org/10.1109/TSE.2002.1000452

EICK, S.G., GRAVES, T.L., KARR, A.F., MARRON, J.S., MOCKUS, A., 1999. **"Does Code Decay? Assessing the Evidence from Change Management Data"**. IEEE Transactions on Software Engineering, Volume 27, Issue 1, pp 1 - 12.

ENGLISH, M. EXTON, C., RIGON, I., CLEARY, B. 2009. **"Fault detection and prediction in an open-source software project"**. Proceedings of the 5th International Conference on Predictor Models in Software Engineering, May 18-19, 2009, Vancouver, British Columbia, Canada. DOI= http://doi.acm.org/10.1145/1540438.1540462

FENTON, N.E., NEIL, M., MARSH, W., HEARTY, P., RADLINSKI, L., KRAUSE, P. 2008. **"On the effectiveness of early life cycle defect prediction with Bayesian Nets".** In *Proceedings of Empirical Software Engineering*. 2008, 499-537. DOI= http://dx.doi.org/10.1007/s10664-008-9072-x

FERNELEY, E. H. 1999. **"Design metrics as an aid to software maintenance: an empirical study".** *Journal of Software Maintenance* 11, 1 (January 1999), 55-72.

DOI= http://dx.doi.org/10.1002/(SICI)1096-908X(199901/02)11:1%3C55::AID-SMR184%3E3.3.CO;2-F

FERRUCCI, F., GRAVINO, C., and DI MARTINO, S.. 2008. **"A Case Study Using Web Objects and COSMIC for Effort Estimation of Web Applications".** In *Proceedings of the 2008 34th Euromicro Conference Software Engineering and Advanced Applications* (SEAA '08). IEEE Computer Society, Washington, DC, USA, 441-448. DOI=http://dx.doi.org/10.1109/SEAA.2008

FORRESTER, J.W., (1961), **"Industrial Dynamics"**, Cambridge, MA: The MIT Press.

GALL, H., JAZAYERI, M., KLOSCH, R.R., TRAUSMUTH, G., 1997, **"Software Evolution Observations Based on Product Release History"**. In: Proc. 1997 Intern. Conf. Software Maintenance (ICSM´97).

HEIJSTEK, W., CHAUDRO, M.R.V. 2009. **"Empirical Investigations of Model Size, Complexity and Effort in a Large Scale, Distributed Model Driven Development Process".** Software Engineering and Advanced Applications, 2009. SEAA '09. 35th Euromicro Conference on 2009 , Page(s): 113 – 120. DOI=http://dx.doi.org/10.1109/SEAA.2009.70.

IEEE, 1990. **IEEE Standard Glossary of Software Engineering Terminology.** Disponível em: http://www.mendeley.com/research/ieee-610standard-glossary-of-software-engineering-terminology.

ISO 9126-1, 1997, **International Standard. Information Technology – Software Quality Characteristics and Metrics** – Part 1: Quality Characteristics and Sub-Characteristics.

ISO/IEC. 2003. **"Software engineering--Software product quality. Part 3: Internal metrics".** ISO/IEC TR 9126-3, Geneva, Switzerland, International Organization for Standardization.

ISO/IEC 12207, 2008. **"Systems and software engineering – Software life cycle processes".**

JABREF. 2011. **"JabRef Reference Manager"**. Disponível em: http://jabref.sourceforge.net/. Acessado em: Maio, 2011.

KELLNER, M.I., MADACHY, R.J., RAFFO, D. 1999. **"Software process simulation modeling: Why? What? How?".** In Proceedings of Journal of Systems and Software. 1999, 91-105.

KEMERER, C.F., SLAUGHTER, S., 1999. **"An Empirical Approach to Studying Software Evolution"**. IEEE Transactions on Software Engineering, Volume 25, Issue 4, pp 493 – 509, ISSN 0098-5589.

KITCHENHAM, B., (2004). **"Procedures for Performing Systematic Reviews"**, Joint Technical Report Keele University TR/SE-0401 and NICTA Technical Report 0400011T.1, Keele University and NICTA.

KIRKWOOD, C. W. 1998. **"System Dynamics Methods: A Quick Introduction".**www.public.asu.edu/~kirkwood/sysdyn/SDIntro/SDIntro.htm Acessado em 03/05/2011.

KNOP, I. O. 2009. **"Infraestrutura para Simulação de Processos de Software Baseada em Metamodelos de Dinâmica de Sistemas".** Dissertação de Mestrado. UFJF – Universidade Federal de Juiz de Fora.

KOZLOV D., KOSKINEN J., SAKKINEN M., MARKKULA J. 2008. **"Assessing maintainability change over multiple software releases".** In Journal of Software Maintenance and Evolution: Research and Practice 2008; 20:31–58. DOI= http://dx.doi.org/10.1002/smr.v20:1

LEHMAN, M.M. ,1980. **"Programs, Life Cycle and the Laws of Software Evolution"**, Proc. IEEE Special Issue on Software Engineering, vol. 68, no. 9, pp. 1060 -1076.

LEHMAN, M.M., RAMIL, J.F., WERNICK, P. D., PERRY, D. E., TURSKI, W. M. 1997. **"Metrics and Laws of Software Evolution - The Nineties View"**. In *Proceedings of the 4th International Symposium on Software Metrics* (METRICS '97). IEEE Computer Society, Washington, DC, USA, 20-.

LEHMAN, M.M., PERRY, D.E., RAMIL, J.F., 1998, **"Implications of Evolution Metrics on Software Maintenance".** Proceedings International Conference on Software Maintenance, Volume 16, Issue 20, pp 208 – 217.

LEHMAN, RAMIL, KAHEN, 2000. **"Evolution as a Noun and Evolution as a Verb."** SOCE 2000 Workshop on Software and Organization Co-evolution, July 13 - 14, 2000.

LEHMAN, M.M., RAMIL, J.F. 2001. **"Rules and Tools for Software Evolution Planning and Management"**. In: Annals of Software Engineering, November, vol. 11, no. 1, (2001) pp. 15-44(30).

LEHMAN M.M., RAMIL, J.F. 2002. **"Software Evolution and Software Evolution Processes"**, Annals of Software Engineering, special issue on Software Process-based Software Engineering, vol. 14, 2002, pp. 275–309.

LEHMAN M.M., RAMIL J.F. 2003. **"Software evolution: background, theory, practice."** Information Processing Letters, v. 88 n. 1-2, p. 33-44, 2003.

LESZAK, M. 2005. **"Software Defect Analysis of a Multi-release Telecommunications System".** *In Product Focused Software Process Improvement* (2005), pp. 98-114.

MADHAVJI, RAMIL, PERRY, 2006. **"Software Evolution and Feedback: Theory and Practice."** John Wiley & Sons, 2006.

MADACHY, R.J. 2008. **"Software process dynamics."** Wiley, IEEE Press, Hoboken, Piscataway (2008).

MUELLER, M., PFAHL, D. 2008. **Simulation Methods**, In: Guide to Advanced Empirical Software Engineering, ed. by Forrest Shull, Janice Singer, Dag I.K. Sjøberg. Springer-Verlag London, chap. 5, pp. 117-152. (ISBN: 13:978-1-84800-043-8), 2008.

PAI M., MCCULLOCH M., GORMAN J., PAI N., ENANORIA W., KENNEDY G. 2004. **"Systematic reviews and meta-analysis: an illustrated step-by-step guide".** The National medical Journal of India. 2004;17:86–95.

PFLEEGER, S.L., 1998**, "The Nature of System Modification".** IEEE Software.

PRESSMAN, R. S. 2009. **"Software Engineering: A Practitioner's Approach"**, McGraw Hill, 7th ed, New York.

POELS, G., DEDENE, G. 2001. **"Evaluating the Effect of Inheritance on the Modifiability of Object-Oriented Business Domain Models".** In Proceedings of the Fifth European Conference on Software Maintenance and Reengineering (CSMR '01). IEEE Computer Society, Washington, DC, USA.

RAFFO, D. M., VANDEVILLE, J. V., MARTIN, R. H. 1999. **"Software Process Simulation to Achieve Higher CMM Levels".** Journal of Systems and Software, Number 6, 1999.

RIAZ, M.; MENDES, E.; TEMPERO, E. A. 2009. **"Systematic Review of Software Maintainability Prediction and Metrics"**. *Proceeding ESEM '09 Proceedings of the* 2009 3rd International Symposium on Empirical Software Engineering and

Measurement. IEEE Computer Society, Washington, DC, USA, 367-377. DOI= http://dx.doi.org/10.1109/ESEM.2009.5314233

SCACCHI, W., 2003. **"Understanding Open Source Software Evolution: Applying, Breaking, and Rethinking the Laws of Software Evolution"**, disponível em http://www.ics.uci.edu/~wscacchi/Papers/New/Understanding-OSS-Evolution.pdf, acessado em 11 de abril de 2011.

SCHNEIDEWIND, N., HINCHEY, M. 2009. **"A Complexity Reliability Model".** In *Proceedings of the 20th IEEE international conference on software reliability engineering* (ISSRE'09) on. page(s): 1 – 10.

SENTAS, P., ANGELIS, L., STAMELOS, I. 2008. **"A Statistical Framework for Analyzing the Duration of Software Projects".** *Empirical Software Engineering (Springer)*. 13:147–184. DOI=http://dx.doi.org/10.1007/s10664-007-9051-7

SHANNON, R. E. 1998. **"Introduction to the art and science of simulation"** Proceedings of the 30th conference on Winter simulation, p.7-14, December 13-16, Washington, D.C., United States.

STOPFORD, B., COUNSELL, S., 2008, **"A Framework for the Simulation of Structural Software Evolution"**. ACM Transactions on Modeling and Computer Simulation, Vol. 18, No. 4, Article 17.

THOMAS, L.G., SCHACH, S.R., HELLER, G.Z., OFFUTT, J., 2009, **"Impact of release intervals on empirical research into software evolution, with application to the maintainability of Linux"**. IET Software, Vol. 3, Iss. 1, pp. 58-66

TRAVASSOS, G.H., SHULL, F., CARVER, J., 2001, **"Working with UML: A Software Design Process Based on Inspections for the Unified Modeling Language"**, Advances in Computers, San Diego, v.54, n.1, p.35 - 97.

TRAVASSOS, G.H., BARROS, M.O., 2003, **"Contributions of In Virtuo and In Silico Experiments for the Future of Empirical Studies in Software Engineering"**, In: Proc. of the WSESE03, Fraunhofer IRB Verlag, Rome.

TRAVASSOS, G. H., SANTOS, P. S. M., MIAN, P., DIAS NETO, A. C.,BIOLCHINI, J. 2008. **An Environment to Support Large Scale Experimentation in Software Engineering**. In: Proc. of the IEEE International Conference on Engineering of Complex Computer Systems, ICECCS 2008, p. 193-202.

VENSIM. 2003a.**Vensim 5 Reference Manual**, Ventana Systems, Inc., January 25, 2003.

VENSIM. 2003b. **Vensim 5 Modeling Guide**, Ventana Systems, Inc., January 25, 2003

VENSIM, 2011. **Vensim PLE - Personal Learning Edition**, Ventana Systems, Inc., Disponível em: http://www.vensim.com/freedownload.html. Acessado em: Maio, 2011.

YUEN, C.K., 1985, **"An Empirical Approach to the Study of Errors in Large Software under Maintenance"**. In: Proc. Second Conf. Software Maintenance, pp. 96-105, ISBN 0-8186-0648-7.

YUEN, C.K., 1987, **"A Statistical Rationale for Evolution Dynamic Concepts"**. In: Proc. Conf. Software Maintenance, IEEE.

YUEN, C.K., 1988, **"On Analyzing Maintenance Process Data at the Global and Detailed Levels: A Case Study"**. In: Proc. Fourth Conf. Software Maintenance, IEEE, pp. 248-255.

ZHANG, H., KITCHENHAM, B., PFAHL, D. 2008a. **"Reflections on 10 Years of Software Process Simulation Modeling: A Systematic Review"** in International Conference on Software Process (ICSP) Leipzig: Springer.

ZHANG, J. 2008b. **"The Establishment and Application of Effort Regression Equation".** In *Proceedings of the 2008 International Conference on Computer Science and Software Engineering - Volume 02* (CSSE '08), Vol. 2. IEEE Computer Society, Washington, DC, USA, 11-14. DOI= http://dx.doi.org/10.1109/CSSE.2008.726

ZHANG, H. 2009. **"An investigation of the relationships between lines of code and defects".** ICSM, pp.274-283, 2009 IEEE International Conference on Software Maintenance, 2009..

APÊNDICE A – *quasi* Revisão Sistemática

A.1 Introdução

Esse apêndice apresenta os detalhes da reexecução do estudo secundário, baseado em quasi revisão sistemática, realizado no contexto dessa dissertação. O estudo foi executado em outubro de 2010, nas máquinas de busca *EI Compendex e SCOPUS.*

Na reexecução do estudo a nova data considerada, para a máquina de busca *EI Compendex*, usada no protocolo original, coomprende o perído entre 2007 e 2010, já que anteriormente o estudo realizado considerou os artigos publicados até o final de 2006. Dentre as fontes de publicação indexadas pela *EI Compendex* estão; *IEEE* e *ACM.*

Para ampliar a abrangência do estudo e dar maior confiabilidade aos resultados apresentados a máquina de busca *SCOPUS* foi incluída como nova fonte de pesquisa. Ela indexa os artigos das principais fontes de publicação da área de Engenharia de Software, dentre elas, *IEEE, ACM, Cite Seerx, Elsevier e Springer*. Para a nova máquina de busca incluída no protocolo foi necessário realizar a busca sem restrição de data, ou seja, buscar em artigos publicados até a data atual, outubro de 2010.

A.2 Questões de Pesquisa

As questões de pesquisa representam a base para identificar a influência entre as características de software, dependendo da etapa do processo de desenvolvimento de software. De acordo com ARAÚJO (2009), neste trabalho as seguintes etapas do processo de desenvolvimento de software foram consideradas; Especificação de Requisitos, Projeto de Alto e Baixo Nível e Codificação, por estarem usualmente presentes no desenvolvimento e manutenção de software orientados a objetos.

Q01: Avaliação da influência entre Tamanho e Complexidade
Q02: Avaliação da influência entre Tamanho e Confiabilidade
Q03: Avaliação da influência entre Complexidade e Esforço
Q04: Avaliação da influência entre Esforço e Confiabilidade
Q05: Avaliação da influência entre Complexidade e Manutenibilidade
Q06: Avaliação da influência entre Esforço e Manutenibilidade
Q07: Avaliação da influência entre Esforço e Periodicidade
Q08: Avaliação da influência entre Periodicidade e Manutenibilidade

Q09: Avaliação da influência entre Tamanho e Esforço

Q10: Avaliação da influência entre Tamanho e Manutenibilidade

Q11: Avaliação da influência entre Periodicidade e Tamanho

Q12: Avaliação da influência entre Complexidade e Confiabilidade

Q13: Avaliação da influência entre Periodicidade e Complexidade

Q14: Avaliação da influência entre Manutenibilidade e Confiabilidade

Q15: Avaliação da influência entre Periodicidade e Confiabilidade

2.1. Formulação das Questões

P1: Existe alguma Influência entre a característica de software <A> e <B> na etapa <X> do processo de desenvolvimento de software orientados a objetos?

1. **Problema:** encontrar trabalhos de pesquisa que identificam a influência entre as características de software <A> e <B>.
2. **Intervenção:** influência entre as características de software <A> e <B>.
3. **Controle:** não definido.
4. **Efeito:** caracterização da influência entre as características de software <A> e <B>
5. **Resultados:** número de trabalhos encontrados que identificam influência entre as características de software <A> e <B>.
6. **População:** resultados de estudos primários relacionados a projetos de software orientados a objetos que descrevem a influência entre as características de software <A> e <B>.
7. **Aplicação:** fundamental para evidenciar a influência entre as características de software <A> e <B>, para dar suporte ao modelo de causas e efeitos construído com bases nas Leis de Evolução de Software.

Com os resultados obtidos na questão P1, iremos procurar por respostas para a segunda questão:

P1.1 Qual a direção da influência entre as características de software <A> e <B> na etapa <X> do processo de desenvolvimento de software orientados a objetos?

1. **Problema:** encontrar trabalhos de pesquisa que identificam a direção da influência entre as características de software <A> e <B>.

2. **Intervenção:** direção da influência entre as características de software <A> e <B>.
3. **Controle:** não definido.
4. **Efeito:** caracterização da direção da influência entre as características de software <A> e <B>
5. **Resultados:** número de trabalhos encontrados que identificam qual a direção da influência entre as características de software <A> e <B>.
6. **População:** artigos selecionados na questão P1
7. **Aplicação:** fundamental para avaliar a direção da influência entre as características de software <A> e <B>, para dar suporte ao modelo de causas e efeitos construído com bases nas Leis de Evolução de Software.

Com os resultados obtidos na questão P1.1, iremos procurar por respostas para a terceira questão:

P1.1.1 Qual a intensidade/taxa da influência entre as características de software <A> e <B> na etapa <X> do processo de desenvolvimento de software orientados a objetos?

1. **Problema:** encontrar trabalhos de pesquisa que identificam a intensidade/taxa da influência entre as características de software <A> e <B>.
2. **Intervenção:** intensidade/taxa da influência entre as características de software <A> e <B>.
3. **Controle:** não definido.
4. **Efeito:** caracterização da intensidade/taxa da influência entre as características de software <A> e <B>
5. **Resultados:** número de trabalhos encontrados que identificam qual a intensidade/taxa da influência entre as características de software <A> e <B>.
6. **População:** artigos selecionados na questão P1.1

Aplicação: fundamental para avaliar a intensidade/taxa da influência entre as características de software <A> e <B>, para dar suporte ao modelo de causas e efeitos construído com bases nas Leis de Evolução de Software

2.2. Seleção de Fontes

A fonte básica de informação será representada por algumas bibliotecas digitais, incluindo conferências e publicações listadas abaixo. Também será considerada a busca por processo de conferências, cujos temas estão relacionados com a manutenção de software.

2.3. Palavras Chaves Gerais

- Software caracteristics
- Metric
- Relation, relationship, correlation, dependency, influence, effort (específicas para P1)
- Direction, primary study, experimental study, empirical study (específicas para P1.1)
- Intensity, rate (específicas para P1.1.1)

2.4. Idioma dos Trabalhos

Inglês

2.5. Identificação de Fontes

2.5.1. Métodos para seleção de fontes

As fontes de busca devem ser acessadas via web. No contexto desta revisão, a busca manual não deve ser inicialmente considerada.

2.5.2. Lista das Máquinas de Busca

- Scopus
- EI Compendex

2.5.3. Tipos de Artigos

Teóricos, provas de conceito, estudos experimentais

2.5.4. Critérios de Inclusão e Exclusão de Artigos

- Os artigos devem estar disponíveis na internet;
- Os artigos devem estar escritos em Inglês;
- Os artigos devem considerar estudos do relacionamento entre métricas ou características de software;

- Os artigos devem considerar aplicações de software do tipo *E-type* Systems, que são sistemas ativamente usados e embutidos em um domínio do mundo real (LEHMAN e RAMIL 2003). Portanto, não serão considerados estudos relacionados a sistemas de software considerados de baixo nível (sistemas operacionais, compiladores, protocolos);
- Os artigos devem estar relacionados com a influência entre as características de software (para a questão P1);
- Os artigos devem estar relacionados com a direção da influência entre as características de software (para a questão P1.1);
- Os artigos devem estar relacionados com a intensidade/taxa da influência entre as características de software (para a questão P1.1.1);

2.5.5. Processo de Seleção Preliminar de Estudos

Um pesquisador irá aplicar a estratégia de busca para identificar os estudos em potencial. Os estudos identificados serão selecionados pelo mesmo pesquisador através da leitura e da verificação dos critérios de inclusão e exclusão estabelecidos, o que inclui a extração de informações. Uma vez realizado esta etapa, um segundo pesquisador irá avaliar os resultados, para entrar em consenso sobre os estudos selecionados.

2.6. Estratégia de Extração de Informações

Para cada estudo selecionado, depois da execução do processo de seleção, o pesquisador irá extrair as seguintes informações.

- Título do Estudo
- Autores
- Fonte de publicação
- Data de publicação
- Tipo do Estudo
- Categoria
- Contexto e tecnologias aplicadas
- Etapas do Desenvolvimento Envolvidas
- Lista de métricas e características de software
- Descrição da influência entre as métricas ou características de software:
- Descrição da direção da influência entre as métricas ou características de software

- Descrição da intensidade da influência entre as métricas ou características de software

A informação sobre as etapas do desenvolvimento envolvidas no artigo analisado foi adicionada na estratégia de extração de informações para dar suporte sobre a aplicabilidade do modelo nas etapas do desenvolvimento de software considerdas neste trabalho.

2.7. *Strings* de Busca

Devido ao fato de o ambiente de busca ser extenso, será necessário restringir o escopo da busca. Esta restrição varia de acordo com a questão de busca utilizada e a consideração de onde as palavras chaves serão pesquisadas, no texto inteiro ou apenas no resumo dos trabalhos.

A estrutura das *strings* de busca seguem um determinado padrão para as questões que estão relacionadas à mesma etapa de desenvolvimento e também um padrão geral.

O padrão geral segue a estrutura abaixo.

(Palavras chaves gerais que determinam o problema a ser estudado)
AND
(Palavras chaves relacionadas à característica de software A)
AND
(Palavras chaves relacionadas à característica de software B)
AND
(Palavras chaves relacionadas à etapa do processo de desenvolvimento de software X)
AND
(Palavras chaves relacionadas à população a ser analisada)

Portanto, as *strings* de busca construídas terão uma estrutura muito semelhante e isso afetará o resultado da execução das *strings* nas máquinas, no sentido dos resultados para diferentes questões retornarem artigos duplicados.

2.8. Resumo dos Resultados

Os resultados encontrados serão apresentados utilizando as tabelas apresentadas abaixo.

Tabela 1 - Para a questão P1 (influência entre as características de software <A> e <B>):

Artigo	Sim	Não	Não Conclusivo

Tabela 2 - Para a questão P1.1 (direção da influência entre as características de software <A> e <B>):

Artigo	A →B	B →A	Não Conclusivo

Tabela 3 - Para a questão P1.1.1 (intensidade/taxa da influência entre as características de software <A> e <B>):

Artigo	Taxa	Não Conclusivo

2.9. Avaliação da Qualidade dos Estudos Primários

Cada artigo deve ser avaliado de acordo com as questões da tabela 4, atribuindo uma pontuação de qualidade por artigo.

Tabela 4 – Questões para avaliação da qualidade dos artigos selecionados

Questão	Pontuação de Qualidade por Artigo				
	Artigo 1	Artigo 2	Artigo 3	Artigo 4	Artigo 5
1. Os dados foram analisados apropriadamente?					
1.1. Os dados foram analisados para identificar *outliers* e para avaliar as propriedades da distribuição antes da análise?					
1.2. Os resultados da análise foram utilizados apropriadamente para transformar os dados e para selecionar os *data points* apropriados?					
2. O estudo realizou análise de sensibilidade ou residual?					
2.1 Os modelos de estimativa sofreram análise de sensibilidade ou residual?					
2.2 Os resultados da análise de sensibilidade ou residual foram utilizados para remover dados anormais quando necessário?					

3. Foram utilizados métodos estatísticos adequados com base na escala de dados brutos e o tipo de distribuição?					
4. Quão bons foram os métodos de comparação utilizados no estudo?					
5. Estão definidos ou indicados quais foram os projetos utilizados para construir cada modelo?					
6. Está descrito ou sugerido como a precisão do modelo (erro estimativa) foi medida?					
7. Está explicito qual o método de validação cruzada foi utilizado?					
8. Todos os métodos do constructo (modelo) foram totalmente definidos (ferramentas e métodos usados)?					
Total da pontuação dos artigos primários					

As questões da tabela acima que não possuírem sub-questões serão respondidas SIM/NÃO, correspondendo à pontuação 1 e 0, respectivamente. Quando uma questão possuir sub-questões, a pontuação para cada sub-questão será atribuída de tal forma que a pontuação total da questão principal estará entre 1 e 0.

Para a questão 4 da tabela anterior, o seguinte critério será utilizado: Menos de 2 projetos = Qualidade ruim (pontuação = 0); Entre 2 e 5 projetos = Qualidade razoável (pontuação 0.33); Entre 5 e 10 projetos = Qualidade boa (pontuação = 0.67); Mais de 10 projetos = Qualidade Excelente (pontuação = 1).

A pontuação das questões 5 e 8 variam entre 1 e 0, representando uma qualidade muito ruim e uma excelente qualidade, respectivamente.

As questões 1 e 3 foram definidas como sendo as mais importantes para avaliação da qualidade dos artigos e por isso tem um peso igual à 2, em quanto, as demais questões foram classificadas como um adicional à qualidade dos artigos e portanto não obrigatórias, com um peso igual a 1.

De acordo com os critérios de avaliação da qualidade dos artigos definidos, a pontuação máxima que um artigo pode receber é de 10 pontos.

O pesquisador irá avaliar cada artigo de acordo com cada critério. Pontuações moderadas serão atribuídas aos estudos primários e serão apresentadas na tabela, e indicarão, de acordo como o esquema de pontuação, os artigos com pontuação mais alta ou mais baixa. Em caso de divergência entre os artigos, aqueles com a maior pontuação deverão ser considerados. Com base nos critérios de pontuação definidos, os artigos que obtiverem uma pontuação inferior a 4 pontos deverão ser desconsiderados.

A.3 Execução das Strings de Busca

3.1. Configuração das Máquinas de Busca

Nesta seção serão apresentadas as configurações utilizadas para as máquinas de busca e também todas as *strings* de busca utilizadas para cada questão de pesquisa e para cada etapa do processo do desenvolvimento de software.

3.1.1. Scopus

Nesta máquina de busca as *strings* foram executadas no modo *Advanced Search* e os resultados foram refinados utilizando o filtro *Subject Area*, selecionado a opção *Computer Science.* Não houve restrição de data para o ano de publicação dos artigos.

3.1.2. EI Compendex

Nesta máquina de busca as *strings* foram executadas no modo *Expert Search* e a opção *Autostemming* ligada, para dar maior abrangência na pesquisa, já que o *Autostemming* considera variações das palavras que não estão entre chave. Como para esta base a busca foi executada como uma atualização houve restrição na data de publicação dos artigos, considerando os artigos publicados entre 2007 e 2010.

3.1.3. Strings de Busca – Scopus e EI Compendex

A estrutura das *strings* de busca foi a mesma para as duas máquinas de busca, á que elas possuem a mesma sintaxe de construção das *strings.* Portanto, abaixo serão listadas as *strings* de busca com estrutura exata com que foram executadas em ambas as máquinas.

Para a questão **P1: Q1 - Avaliação da influência entre Tamanho e Complexidade**

String associada à etapa de Especificação de Requisitos

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) AND (size OR {Function Points} OR {Use Case Points} OR {Requirement}) AND (complexity OR {use case}) AND ({Requirement Specification} OR {Requirement Elicitation} OR {Requirement Definition} OR Analysis OR {User Requirement} OR {Requisite}) AND ({Software Characteristic} OR {Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

String associada à etapa de Projeto de Alto Nível

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) AND (size OR Classes OR {Methods per Class}) AND (complexity OR {Class Diagrams} OR {Sequence Diagrams} OR {State Diagrams} OR {Package Diagrams} or {Activity Diagrams} OR {Depth of Inheritance per Class} OR {Depth of Inheritance Tree} OR DIT OR {Children per Class} OR NOC OR {Number of Children}) AND (Design OR {High Level Design} OR Analysis) AND ({Software Characteristic} OR {Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

String associada à etapa de Projeto de Baixo Nível

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) AND (size OR {Key Classes} OR {Support Classes} OR {Methods per Class} OR {Subsystems}) AND (complexity OR {Class Diagrams} OR {Sequence Diagrams} OR {Depth of Inheritance per Class} OR {Depth of Inheritance Tree} OR DIT OR {Coupling between Objects} OR CBO OR {Response for a Class} OR RFC OR {Lack of Cohesion in Methods} OR LCOM OR {Children per Class} OR NOC OR {Number of Children}) AND (Design OR {Low Level Design} OR {Detailed Design}) AND ({Software Characteristic} OR {Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

String associada à etapa de Codificação

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) AND (size OR LOC OR {lines of code} OR {source lines of code} OR {methods per classes}) AND (complexity OR {Depth of Inheritance per Class} OR {Coupling between Objects} OR {Response for a Class} OR {Lack of Cohesion in Methods} OR {Children per Class} OR {Cyclomatic Complexity per Method}) AND (Codification OR Programming OR Building OR Construction OR Implementation OR Coding) AND ({Software Characteristic} OR {Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

Tabela 5 - Quantidade de Artigos Retornados para Q1

Etapa	# Artigos	
	Scopus (até 2010)	EI Compendex (2007 – 2010)
Especificação de Requisitos	174	19
Projeto de Alto Nível	228	21
Projeto de Baixo Nível	116	9
Codificação	131	15

Para a questão **P1: Q2 - Avaliação da influência entre Tamanho e Confiabilidade**

String associada à etapa de Especificação de Requisitos

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) AND (size OR {Function Points} OR {Use Case Points} OR {Requirement}) AND (reliability OR {Detected Defects} OR {Corrected Defects}) AND ({Requirement Specification} OR {Requirement Elicitation} OR {Requirement Definition} OR Analysis OR {User Requirement} OR {Requisite}) AND ({Software Characteristic} OR {Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

String associada à etapa de Projeto de Alto Nível

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) AND (size OR Classes OR {Methods per Class})

AND (reliability OR {Detected Defects} OR {Corrected Defects}) AND (Design OR {High Level Design} OR Analysis) AND ({Software Characteristic} OR {Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

String associada à etapa de Projeto de Baixo Nível

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) AND (size OR {Key Classes} OR {Support Classes} OR {Methods per Class} OR {Subsystems}) AND (reliability OR {Detected Defects} OR {Corrected Defects}) AND (Design OR {Low Level Design} OR {Detailed Design}) AND ({Software Characteristic} OR {Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

String associada à etapa de Codificação

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) AND (size OR LOC OR {lines of code} OR {source lines of code} OR {methods per classes}) AND (reliability OR {Detected Defects} OR {Corrected Defects} OR {System Availability}) AND (Codification OR Programming OR Building OR Construction OR Implementation OR Coding) AND ({Software Characteristic} OR {Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

Tabela 6 - Quantidade de Artigos Retornados para Q2

Etapa	# Artigos	
	Scopus (até 2010)	EI Compendex (2007 - 2010)
Especificação de Requisitos	137	10
Projeto de Alto Nível	155	10
Projeto de Baixo Nível	81	4
Codificação	96	8

Para a questão **P1: Q3 - Avaliação da influência entre Complexidade e Esforço**

String associada à etapa de Especificação de Requisitos

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) AND (complexity OR {use case}) AND (effort OR {Requirements handled} OR {Use Cases handled} OR efficiency OR {People} OR {Allocated Resources} OR {Spent Time} OR {Average Productivity}) AND ({Requirement Specification} OR {Requirement Elicitation} OR {Requirement Definition} OR Analysis OR {User Requirement} OR {Requisite}) AND ({Software Characteristic} OR {Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

String associada à etapa de Projeto de Alto Nível

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) AND (complexity OR {Class Diagrams} OR {Sequence Diagrams} OR {State Diagrams} OR {Package Diagrams} or {Activity Diagrams} OR {Depth of Inheritance per Class} OR {Depth of Inheritance Tree} OR DIT OR {Children per Class} OR NOC OR {Number of Children})AND (effort OR {Class diagrams handled} OR {Sequence diagrams handled} OR {State Diagrams handled} OR {Package Diagrams

handled} OR {Activity Diagrams handled} OR efficiency OR {People} OR {Allocated Resources} OR {Spent Time} OR {Average Productivity}) AND (Design OR {High Level Design} OR Analysis) AND ({Software Characteristic} OR {Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

String associada à etapa de Projeto de Baixo Nível

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) AND (complexity OR {Class Diagrams} OR {Sequence Diagrams} OR {Depth of Inheritance per Class} OR {Depth of Inheritance Tree} OR DIT OR {Coupling between Objects} OR CBO OR {Response for a Class} OR RFC OR {Lack of Cohesion in Methods} OR LCOM OR {Children per Class} OR NOC OR {Number of Children}) AND (effort OR {Class diagrams handled} OR {Sequence diagrams handled} OR efficiency OR {People} OR {Allocated Resources} OR {Spent Time} OR {Average Productivity}) AND (Design OR {Low Level Design} OR {Detailed Design}) AND ({Software Characteristic} OR {Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

String associada à etapa de Codificação

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) AND (complexity OR {Depth of Inheritance per Class} OR {Coupling between Objects} OR {Response for a Class} OR {Lack of Cohesion in Methods} OR {Children per Class} OR {Cyclomatic Complexity per Method}) AND (effort OR {LOC handled} OR {lines of code handled} OR {source lines of code handled} OR efficiency OR {People} OR {Allocated Resources} OR {Spent Time} OR {Average Productivity}) AND (Codification OR Programming OR Building OR Construction OR Implementation OR Coding) AND ({Software Characteristic} OR {Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

Tabela 7 - Quantidade de Artigos Retornados para Q3

Etapa	# Artigos	
	Scopus (até 2010)	EI Compendex (2007 - 2010)
Especificação de Requisitos	234	28
Projeto de Alto Nível	242	41
Projeto de Baixo Nível	190	30
Codificação	193	25

Para a questão **P1: Q4 - Avaliação da influência entre Esforço e Confiabilidade**

String associada à etapa de Especificação de Requisitos

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) AND (effort OR {Requirements handled} OR {Use Cases handled} OR efficiency OR {People} OR {Allocated Resources} OR {Spent Time} OR {Average Productivity}) AND (reliability OR {Detected Defects} OR {Corrected Defects}) AND ({Requirement Specification} OR {Requirement Elicitation} OR {Requirement Definition} OR Analysis OR {User Requirement} OR {Requisite}) AND ({Software Characteristic} OR {Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

String associada à etapa de Projeto de Alto Nível

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) AND (effort OR {Class diagrams handled} OR {Sequence diagrams handled} OR {State Diagrams handled} OR {Package Diagrams handled} OR {Activity Diagrams handled} OR efficiency OR {People} OR {Allocated Resources} OR {Spent Time} OR {Average Productivity}) AND (reliability OR {Detected Defects} OR {Corrected Defects}) AND (Design OR {High Level Design} OR Analysis) AND ({Software Characteristic} OR {Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

String associada à etapa de Projeto de Baixo Nível

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) AND (effort OR {Class diagrams handled} OR {Sequence diagrams handled} OR efficiency OR {People} OR {Allocated Resources} OR {Spent Time} OR {Average Productivity}) AND (reliability OR {Detected Defects} OR {Corrected Defects}) AND (Design OR {Low Level Design} OR {Detailed Design}) AND ({Software Characteristic} OR {Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

String associada à etapa de Codificação

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) AND (effort OR {LOC handled} OR {lines of code handled} OR {source lines of code handled} OR efficiency OR {People} OR {Allocated Resources} OR {Spent Time} OR {Average Productivity}) AND (reliability OR {Detected Defects} OR {Corrected Defects} OR {System Availability}) AND (Codification OR Programming OR Building OR Construction OR Implementation OR Coding) AND ({Software Characteristic} OR {Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

Tabela 8 - Quantidade de Artigos Retornados para Q4

Etapa	# Artigos	
	Scopus (até 2010)	EI Compendex (2007 – 2010)
Especificação de Requisitos	209	16
Projeto de Alto Nível	217	20
Projeto de Baixo Nível	149	11
Codificação	153	13

Para a questão **P1: Q5 - Avaliação da influência entre Complexidade e Manutenibilidade**

String associada à etapa de Especificação de Requisitos

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) AND (complexity OR {use case}) AND (maintainability OR {Defects} OR {Diagnostic of Defects} OR {Removal of Defects}) AND ({Requirement Specification} OR {Requirement Elicitation} OR {Requirement Definition} OR Analysis OR {User Requirement} OR {Requisite}) AND ({Software Characteristic} OR {Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

String associada à etapa de Projeto de Alto Nível

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) AND (complexity OR {Class Diagrams} OR {Sequence Diagrams} OR {State Diagrams} OR {Package Diagrams} or {Activity Diagrams} OR {Depth of Inheritance per Class} OR {Depth of Inheritance Tree} OR DIT OR {Children per Class}

OR NOC OR {Number of Children}) AND (maintainability OR {Defects} OR {Diagnostic of Defects} OR {Removal of Defects}) AND (Design OR {High Level Design} OR Analysis) AND ({Software Characteristic} OR {Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

String associada à etapa de Projeto de Baixo Nível

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) AND (complexity OR {Class Diagrams} OR {Sequence Diagrams} OR {Depth of Inheritance per Class} OR {Depth of Inheritance Tree} OR DIT OR {Coupling between Objects} OR CBO OR {Response for a Class} OR RFC OR {Lack of Cohesion in Methods} OR LCOM OR {Children per Class} OR NOC OR {Number of Children}) AND (maintainability OR {Defects} OR {Diagnostic of Defects} OR {Removal of Defects}) AND (Design OR {Low Level Design} OR {Detailed Design}) AND ({Software Characteristic} OR {Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

String associada à etapa de Codificação

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) AND (complexity OR {Depth of Inheritance per Class} OR {Coupling between Objects} OR {Response for a Class} OR {Lack of Cohesion in Methods} OR {Children per Class} OR {Cyclomatic Complexity per Method}) AND (maintainability OR {Defects} OR {Diagnostic of Defects} OR {Removal of Defects}) AND (Codification OR Programming OR Building OR Construction OR Implementation OR Coding) AND ({Software Characteristic} OR {Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

Tabela 9 - Quantidade de Artigos Retornados para Q5

Etapa	# Artigos	
	Scopus (até 2010)	EI Compendex (2007 – 2010)
Especificação de Requisitos	158	12
Projeto de Alto Nível	148	15
Projeto de Baixo Nível	151	10
Codificação	130	16

Para a questão **P1: Q6 - Avaliação da influência entre Esforço e Manutenibilidade**

String associada à etapa de Especificação de Requisitos

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) AND (effort OR {Requirements handled} OR {Use Cases handled} OR efficiency OR {People} OR {Allocated Resources} OR {Spent Time} OR {Average Productivity}) AND (maintainability OR {Defects} OR {Diagnostic of Defects} OR {Removal of Defects}) AND ({Requirement Specification} OR {Requirement Elicitation} OR {Requirement Definition} OR Analysis OR {User Requirement} OR {Requisite}) AND ({Software Characteristic} OR {Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

String associada à etapa de Projeto de Alto Nível

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) AND (effort OR {Class diagrams handled} OR {Sequence diagrams handled} OR {State Diagrams handled} OR {Package Diagrams handled} OR {Activity Diagrams handled} OR efficiency OR {People} OR {Allocated Resources} OR {Spent Time} OR {Average Productivity}) AND (maintainability OR {Defects} OR {Diagnostic of Defects} OR {Removal of Defects}) AND (Design OR {High Level Design} OR Analysis) AND ({Software Characteristic} OR {Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

String associada à etapa de Projeto de Baixo Nível

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) AND (effort OR {Class diagrams handled} OR {Sequence diagrams handled} OR efficiency OR {People} OR {Allocated Resources} OR {Spent Time} OR {Average Productivity}) AND (maintainability OR {Defects} OR {Diagnostic of Defects} OR {Removal of Defects}) AND (Design OR {Low Level Design} OR {Detailed Design}) AND ({Software Characteristic} OR {Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

String associada à etapa de Codificação

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) AND (effort OR {LOC handled} OR {lines of code handled} OR {source lines of code handled} OR efficiency OR {People} OR {Allocated Resources} OR {Spent Time} OR {Average Productivity}) AND (maintainability OR {Defects} OR {Diagnostic of effects} OR {Removal of Defects}) AND (Codification OR Programming OR Building OR Construction OR Implementation OR Coding) AND ({Software

Characteristic} OR {Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

Tabela 10 - Quantidade de Artigos Retornados para Q6

Etapa	# Artigos	
	Scopus (até 2010)	EI Compendex (2007 – 2010)
Especificação de Requisitos	137	19
Projeto de Alto Nível	144	25
Projeto de Baixo Nível	113	15
Codificação	113	23

Para a questão **P1: Q7 - Avaliação da influência entre Esforço e Periodicidade**

String associada à etapa de Especificação de Requisitos

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) AND (effort OR {Requirements handled} OR {Use Cases handled} OR efficiency OR {People} OR {Allocated Resources} OR {Spent Time} OR {Average Productivity}) AND (periodicity OR {Interval between Versions}) AND ({Requirement Specification} OR {Requirement Elicitation} OR {Requirement Definition} OR Analysis OR {User Requirement} OR {Requisite}) AND ({Software Characteristic} OR {Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

String associada à etapa de Projeto de Alto Nível

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) AND (effort OR {Class diagrams handled} OR {Sequence diagrams handled} OR {State Diagrams handled} OR {Package Diagrams handled} OR {Activity Diagrams handled} OR efficiency OR {People} OR {Allocated Resources} OR {Spent Time} OR {Average Productivity}) AND (periodicity OR {Interval between Versions}) AND (Design OR {High Level Design} OR Analysis) AND ({Software Characteristic} OR {Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

String associada à etapa de Projeto de Baixo Nível

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) AND (effort OR {Class diagrams handled} OR {Sequence diagrams handled} OR efficiency OR {People} OR {Allocated Resources} OR {Spent Time} OR {Average Productivity}) AND (periodicity OR {Interval between Versions}) AND (Design OR {Low Level Design} OR {Detailed Design}) AND ({Software Characteristic} OR

{Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

String associada à etapa de Codificação

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) AND (effort OR {LOC handled} OR {lines of code handled} OR {source lines of code handled} OR efficiency OR {People} OR {Allocated Resources} OR {Spent Time} OR {Average Productivity}) AND (periodicity OR {Interval between Versions}) AND (Codification OR Programming OR Building OR Construction OR Implementation OR Coding) AND ({Software Characteristic} OR {Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

Tabela 11 - Quantidade de Artigos Retornados para Q7

Etapa	# Artigos	
	Scopus (até 2010)	EI Compendex (2007 – 2010)
Especificação de Requisitos	0	2
Projeto de Alto Nível	0	2
Projeto de Baixo Nível	0	0
Codificação	0	2

Para a questão **P1: Q8 - Avaliação da influência entre Periodicidade e Manutenibilidade**

String associada à etapa de Especificação de Requisitos

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) AND (periodicity OR {Interval between Versions}) AND (maintainability OR {Defects} OR {Diagnostic of Defects} OR {Removal of Defects}) AND ({Requirement Specification} OR {Requirement Elicitation} OR {Requirement Definition} OR Analysis OR {User Requirement} OR {Requisite}) AND ({Software Characteristic} OR {Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

String associada à etapa de Projeto de Alto Nível

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) AND (periodicity OR {Interval between Versions}) AND (maintainability OR {Defects} OR {Diagnostic of Defects} OR {Removal of Defects})

AND (Design OR {High Level Design} OR Analysis) AND ({Software Characteristic} OR {Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

String associada à etapa de Projeto de Baixo Nível

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) ND (periodicity OR {Interval between Versions})
AND (maintainability OR {Defects} OR {Diagnostic of Defects} OR {Removal of Defects}) AND (Design OR {Low Level Design} OR {Detailed Design}) AND ({Software Characteristic} OR {Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

String associada à etapa de Codificação

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) AND (periodicity OR {Interval between Versions}) AND (maintainability OR {Defects} OR {Diagnostic of Defects} OR {Removal of Defects}) AND (Codification OR Programming OR Building OR Construction OR Implementation OR Coding) AND ({Software Characteristic} OR {Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

Tabela 12 - Quantidade de Artigos Retornados para Q8

Etapa	# Artigos	
	Scopus (até 2010)	EI Compendex (2007 – 2010)
Especificação de Requisitos	0	0
Projeto de Alto Nível	0	2
Projeto de Baixo Nível	0	2
Codificação	0	0

Para a questão **P1: Q9 - Avaliação da influência entre Tamanho e Esforço**

String associada à etapa de Especificação de Requisitos

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) AND (size OR {Function Points} OR {Use Case Points} OR {Requirement}) AND (effort OR {Requirements handled} OR {Use Cases handled} OR efficiency OR {People} OR {Allocated Resources} OR {Spent Time} OR {Average Productivity}) AND ({Requirement Specification} OR {Requirement Elicitation} OR {Requirement Definition} OR Analysis OR {User Requirement} OR {Requisite}) AND

({Software Characteristic} OR {Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

String associada à etapa de Projeto de Alto Nível

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) AND (size OR Classes OR {Methods per Class}) AND (effort OR {Class diagrams handled} OR {Sequence diagrams handled} OR {State Diagrams handled} OR {Package Diagrams handled} OR {Activity Diagrams handled} OR efficiency OR {People} OR {Allocated Resources} OR {Spent Time} OR {Average Productivity}) AND (Design OR {High Level Design} OR Analysis) AND ({Software Characteristic} OR {Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

String associada à etapa de Projeto de Baixo Nível

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) AND (size OR {Key Classes} OR {Support Classes} OR {Methods per Class} OR {Subsystems}) AND (effort OR {Class diagrams handled} OR {Sequence diagrams handled} OR efficiency OR {People} OR {Allocated Resources} OR {Spent Time} OR {Average Productivity}) AND (Design OR {Low Level Design} OR {Detailed Design}) AND ({Software Characteristic} OR {Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

String associada à etapa de Codificação

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) AND (size OR LOC OR {lines of code} OR {source lines of code} OR {methods per classes}) AND (effort OR {LOC handled} OR {lines of code handled} OR {source lines of code handled} OR efficiency OR {People} OR {Allocated Resources} OR {Spent Time} OR {Average Productivity}) AND (Codification OR Programming OR Building OR Construction OR Implementation OR Coding) AND ({Software Characteristic} OR {Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

Tabela 13 - Quantidade de Artigos Retornados para Q9

Etapa	# Artigos	
	Scopus (até 2010)	EI Compendex (2007 – 2010)
Especificação de Requisitos	296	27
Projeto de Alto Nível	299	25
Projeto de Baixo Nível	152	9
Codificação	221	26

Para a questão **P1: Q10 - Avaliação da influência entre Tamanho e Manutenibilidade**

String associada à etapa de Especificação de Requisitos

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) AND (size OR {Function Points} OR {Use Case Points} OR {Requirement}) AND (maintainability OR {Defects} OR {Diagnostic of Defects} OR {Removal of Defects}) AND ({Requirement Specification} OR {Requirement Elicitation} OR {Requirement Definition} OR Analysis OR {User Requirement} OR {Requisite}) AND ({Software Characteristic} OR {Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

String associada à etapa de Projeto de Alto Nível

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) AND (size OR Classes OR {Methods per Class}) AND (maintainability OR {Defects} OR {Diagnostic of Defects} OR {Removal of Defects}) AND (Design OR {High Level Design} OR Analysis) AND ({Software Characteristic} OR {Software Metric} OR Software Development Project} OR {software project} OR {software measure} OR {software measurement})

String associada à etapa de Projeto de Baixo Nível

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) AND (size OR {Key Classes} OR {Support Classes} OR {Methods per Class} OR {Subsystems}) AND (maintainability OR {Defects} OR {Diagnostic of Defects} OR {Removal of Defects}) AND (Design OR {Low Level Design} OR {Detailed Design}) AND ({Software Characteristic} OR {Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

String associada à etapa de Codificação

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) AND (size OR LOC OR {lines of code} OR {source lines of code} OR {methods per classes}) AND (maintainability OR {Defects} OR {Diagnostic of Defects} OR {Removal of Defects}) AND (Codification OR Programming OR Building OR Construction OR Implementation OR Coding) AND ({Software Characteristic} OR

{Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

Tabela 14 - Quantidade de Artigos Retornados para Q10

Etapa	# Artigos	
	Scopus (até 2010)	EI Compendex (2007 – 2010)
Especificação de Requisitos	102	7
Projeto de Alto Nível	140	14
Projeto de Baixo Nível	71	6
Codificação	80	9

Para a questão **P1: Q11 - Avaliação da influência entre Periodicidade e Tamanho**

String associada à etapa de Especificação de Requisitos

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) AND (periodicity OR {Interval between Versions}) AND (size OR {Function Points} OR {Use Case Points} OR {Requirement}) AND ({Requirement Specification} OR {Requirement Elicitation} OR {Requirement Definition} OR Analysis OR {User Requirement} OR {Requisite}) AND ({Software Characteristic} OR {Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

String associada à etapa de Projeto de Alto Nível

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) AND (periodicity OR {Interval between Versions}) AND (size OR Classes OR {Methods per Class}) AND (Design OR {High Level Design} OR Analysis) AND ({Software Characteristic} OR {Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

String associada à etapa de Projeto de Baixo Nível

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) AND (periodicity OR {Interval between Versions}) AND (size OR {Key Classes} OR {Support Classes} OR {Methods per Class} OR {Subsystems}) AND (Design OR {Low Level Design} OR {Detailed Design}) AND ({Software Characteristic} OR {Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

String associada à etapa de Codificação

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) AND (periodicity OR {Interval between Versions}) AND (size OR LOC OR {lines of code} OR {source lines of code} OR {methods per classes}) AND

(Codification OR Programming OR Building OR Construction OR Implementation OR Coding) AND ({Software Characteristic} OR {Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

Tabela 15 - Quantidade de Artigos Retornados para Q11

Etapa	# Artigos	
	Scopus (até 2010)	EI Compendex (2007 – 2010)
Especificação de Requisitos	0	3
Projeto de Alto Nível	0	2
Projeto de Baixo Nível	0	1
Codificação	0	2

Para a questão **P1: Q12 - Avaliação da influência entre Complexidade e Confiabilidade**

String associada à etapa de Especificação de Requisitos

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) AND (complexity OR {use case}) AND (reliability OR {Detected Defects} OR {Corrected Defects} OR {System Availability}) AND ({Requirement Specification} OR {Requirement Elicitation} OR {Requirement Definition} OR Analysis OR {User Requirement} OR {Requisite}) AND ({Software Characteristic} OR {Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

String associada à etapa de Projeto de Alto Nível

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) AND (complexity OR {Class Diagrams} OR {Sequence Diagrams} OR {State Diagrams} OR {Package Diagrams} or {Activity Diagrams} OR {Depth of Inheritance per Class} OR {Depth of Inheritance Tree} OR DIT OR {Children per Class} OR NOC OR {Number of Children}) AND (reliability OR {Detected Defects} OR {Corrected Defects} OR {System Availability}) AND (Design OR {High Level Design} OR Analysis) AND ({Software Characteristic} OR {Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

String associada à etapa de Projeto de Baixo Nível

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) AND (complexity OR {Class Diagrams} OR {Sequence Diagrams} OR {Depth of Inheritance per Class} OR {Depth of Inheritance Tree} OR DIT OR

{Coupling between Objects} OR CBO OR {Response for a Class} OR RFC OR {Lack of Cohesion in Methods} OR LCOM OR {Children per Class} OR NOC OR {Number of Children}) AND (reliability OR {Detected Defects} OR {Corrected Defects} OR {System Availability}) AND (Design OR {Low Level Design} OR {Detailed Design}) AND ({Software Characteristic} OR {Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

String associada à etapa de Codificação

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) AND (complexity OR {Depth of Inheritance per Class} OR {Coupling between Objects} OR {Response for a Class} OR {Lack of Cohesion in Methods} OR {Children per Class} OR {Cyclomatic Complexity per Method}) AND (reliability OR {Detected Defects} OR {Corrected Defects} OR {System Availability}) AND (Codification OR Programming OR Building OR Construction OR Implementation OR Coding) AND ({Software Characteristic} OR {Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

Tabela 16 - Quantidade de Artigos Retornados para Q12

Etapa	# Artigos	
	Scopus (até 2010)	EI Compendex (2007 – 2010)
Especificação de Requisitos	152	16
Projeto de Alto Nível	158	20
Projeto de Baixo Nível	128	13
Codificação	112	12

Para a questão **P1: Q13 - Avaliação da influência entre Periodicidade e Complexidade**

String associada à etapa de Especificação de Requisitos

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) AND (periodicity OR {Interval between Versions}) AND (complexity OR {use case}) AND ({Requirement Specification} OR {Requirement Elicitation} OR {Requirement Definition} OR Analysis OR {User Requirement} OR {Requisite}) AND ({Software Characteristic} OR {Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

String associada à etapa de Projeto de Alto Nível

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) AND (periodicity OR {Interval between Versions}) AND (complexity OR {Class Diagrams} OR {Sequence Diagrams} OR {State Diagrams} OR {Package

Diagrams} or {Activity Diagrams} OR {Depth of Inheritance per Class} OR {Depth of Inheritance Tree} OR DIT OR {Children per Class} OR NOC OR {Number of Children}) AND (Design OR {High Level Design} OR Analysis) AND ({Software Characteristic} OR {Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

String associada à etapa de Projeto de Baixo Nível

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) AND (periodicity OR {Interval between Versions}) AND (complexity OR {Class Diagrams} OR {Sequence Diagrams} OR {Depth of Inheritance per Class} OR {Depth of Inheritance Tree} OR DIT OR {Coupling between Objects} OR CBO OR {Response for a Class} OR RFC OR {Lack of Cohesion in Methods} OR LCOM OR {Children per Class} OR NOC OR {Number of Children}) AND (Design OR {Low Level Design} OR {Detailed Design})AND ({Software Characteristic} OR {Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

String associada à etapa de Codificação

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) AND (periodicity OR {Interval between Versions}) AND (complexity OR {Depth of Inheritance per Class} OR {Coupling between Objects} OR {Response for a Class} OR {Lack of Cohesion in Methods} OR {Children per Class} OR {Cyclomatic Complexity per Method}) AND (Codification OR Programming OR Building OR Construction OR Implementation OR Coding) AND ({Software Characteristic} OR {Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

Tabela 17 - Quantidade de Artigos Retornados para Q13

Etapa	# Artigos	
	Scopus (até 2010)	EI Compendex (2007 – 2010)
Especificação de Requisitos	0	4
Projeto de Alto Nível	0	5
Projeto de Baixo Nível	0	3
Codificação	0	3

Para a questão **P1: Q14 - Avaliação da influência entre Manutenibilidade e Confiabilidade**

String associada à etapa de Especificação de Requisitos

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) AND (reliability OR {Detected Defects} OR {Corrected Defects} OR {System Availability}) AND (maintainability OR {Defects} OR {Diagnostic of Defects} OR {Removal of Defects}) AND ({Requirement Specification} OR {Requirement Elicitation} OR {Requirement Definition} OR Analysis OR {User Requirement} OR {Requisite}) AND ({Software Characteristic} OR {Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

String associada à etapa de Projeto de Alto Nível

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) AND (reliability OR {Detected Defects} OR {Corrected Defects} OR {System Availability}) AND (maintainability OR {Defects} OR {Diagnostic of Defects} OR {Removal of Defects}) AND (Design OR {High Level Design} OR Analysis) AND ({Software Characteristic} OR {Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

String associada à etapa de Projeto de Baixo Nível

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) AND (reliability OR {Detected Defects} OR {Corrected Defects} OR {System Availability}) AND (maintainability OR {Defects} OR {Diagnostic of Defects} OR {Removal of Defects}) AND (Design OR {Low Level Design} OR {Detailed Design}) AND ({Software Characteristic} OR {Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

String associada à etapa de Codificação

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) AND (reliability OR {Detected Defects} OR {Corrected Defects} OR

{System Availability}) AND (maintainability OR {Defects} OR {Diagnostic of Defects} OR {Removal of Defects}) AND (Codification OR Programming OR Building OR Construction OR Implementation OR Coding) AND ({Software Characteristic} OR {Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

Tabela 18 - Quantidade de Artigos Retornados para Q14

Etapa	# Artigos	
	Scopus (até 2010)	EI Compendex (2007 – 2010)
Especificação de Requisitos	109	11
Projeto de Alto Nível	110	15
Projeto de Baixo Nível	81	0
Codificação	73	10

Para a questão **P1: Q15 - Avaliação da influência entre Periodicidade e Confiabilidade**

String associada à etapa de Especificação de Requisitos

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) AND (reliability OR {Detected Defects} OR {Corrected Defects} OR {System Availability}) AND (periodicity OR {Interval between Versions}) AND ({Requirement Specification} OR {Requirement Elicitation} OR {Requirement Definition} OR Analysis OR {User Requirement} OR {Requisite}) AND ({Software Characteristic} OR {Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

String associada à etapa de Projeto de Alto Nível

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) AND (reliability OR {Detected Defects} OR {Corrected Defects} OR {System Availability}) AND (periodicity OR {Interval between Versions}) AND (Design OR {High Level Design} OR Analysis) AND ({Software Characteristic} OR {Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

String associada à etapa de Projeto de Baixo Nível

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) AND (reliability OR {Detected Defects} OR {Corrected Defects} OR {System Availability}) AND (periodicity OR {Interval between Versions}) AND (Design OR {Low Level Design} OR {Detailed Design}) AND ({Software Characteristic} OR {Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

String associada à etapa de Codificação

(Relationship OR Relation OR Correlation OR Influence OR Dependence OR Effect OR Linkage) AND (reliability OR {Detected Defects} OR {Corrected Defects} OR {System Availability}) AND (periodicity OR {Interval between Versions}) AND (Codification OR Programming OR Building OR Construction OR Implementation OR Coding) AND ({Software Characteristic} OR {Software Metric} OR {Software Development Project} OR {software project} OR {software measure} OR {software measurement})

Tabela 19 - Quantidade de Artigos Retornados para Q15

Etapa	# Artigos	
	Scopus (até 2010)	EI Compendex (2007 – 2010)
Especificação de Requisitos	0	0
Projeto de Alto Nível	0	0
Projeto de Baixo Nível	0	0
Codificação	0	0

A.4 Análise dos artigos retornados

Os resultados foram organizados na tabele 20 de acordo com os artigos encontrados para cada questão de pesquisa, que também está relacionada com as etapas do processo de desenvolvimento de software considerada neste trabalho.

Tabela 20 – Quantidade total de artigos retornados da execução das *strings*

Questão	Etapa	# Artigos		Total por Questão
		Scopus (até 2010)	EI Compendex (2007 - 2010)	
Q1	Requisitos	174	19	713
	Projeto Alto Nível	228	21	
	Projeto Baixo Nível	116	9	
	Codificação	131	15	
Q2	Requisitos	137	10	501
	Projeto Alto Nível	155	10	
	Projeto Baixo Nível	81	4	
	Codificação	96	8	
Q3	Requisitos	234	28	983
	Projeto Alto Nível	242	41	
	Projeto Baixo Nível	190	30	
	Codificação	193	25	
Q4	Requisitos	209	16	788
	Projeto Alto Nível	217	20	
	Projeto Baixo Nível	149	11	
	Codificação	153	13	
Q5	Requisitos	158	12	641
	Projeto Alto Nível	148	15	
	Projeto Baixo Nível	152	10	
	Codificação	130	16	

Q6	Requisitos	137	19	589
	Projeto Alto Nível	144	25	
	Projeto Baixo Nível	113	15	
	Codificação	113	23	
Q7	Requisitos	0	2	6
	Projeto Alto Nível	0	2	
	Projeto Baixo Nível	0	0	
	Codificação	0	2	
Q8	Requisitos	0	0	4
	Projeto Alto Nível	0	2	
	Projeto Baixo Nível	0	2	
	Codificação	0	0	
Q9	Requisitos	296	27	1055
	Projeto Alto Nível	299	25	
	Projeto Baixo Nível	152	9	
	Codificação	221	26	
Q10	Requisitos	102	7	429
	Projeto Alto Nível	140	14	
	Projeto Baixo Nível	71	6	
	Codificação	80	9	
Q11	Requisitos	0	3	8
	Projeto Alto Nível	0	2	
	Projeto Baixo Nível	0	1	
	Codificação	0	2	
Q12	Requisitos	153	16	612
	Projeto Alto Nível	158	20	
	Projeto Baixo Nível	128	13	
	Codificação	112	12	
Q13	Requisitos	0	4	15
	Projeto Alto Nível	0	5	
	Projeto Baixo Nível	0	3	
	Codificação	0	3	
Q14	Requisitos	109	11	409
	Projeto Alto Nível	110	15	
	Projeto Baixo Nível	81	0	
	Codificação	73	10	
Q15	Requisitos	0	0	0
	Projeto Alto Nível	0	0	
	Projeto Baixo Nível	0	0	
	Codificação	0	0	
TOTAL DE ARTIGOS RETORNADOS		**6085**	**668**	**6753**

É importante ressaltar mais uma vez que, para a máquina de busca *Scopus* não houve restrição de data no momento da busca, pois está máquina não foi considerada na primeira execução deste protocolo por ARAÚJO (2009) e foi incluída no protocolo para dar maior abrangência aos resultados deste trabalho e, portanto, deveria considerar todo o corpo de conhecimento sem a restrição da data, enquanto, para a máquina de busca *EI Compendex*, houve restrição para a data no momento da busca, já que para está máquina foi feita apenas uma atualização considerando o período entre 2007 e 2010.

Em cada uma das máquinas de busca, foram executadas 60 *strings* de busca, já que para cada uma das 15 questões de pesquisa foram criadas 4 *strings* de busca relacionada com cada etapa do processo de desenvolvimento de software.

O total de artigos retornados para as 60 *strings* de busca, considerando as duas máquinas de busca utilizadas, *Scopus* e *EI Compendex,* foi de 6753, entretanto, pelo fato das *strings* de busca terem uma estrutura muito semelhante e muitas vezes as palavras chaves se repetirem em alguma parte das diferentes *strings*, ocorreram muitos artigos duplicados entre o total de artigos retornados.

A tabela 21 mostra a quantidade de artigos retornados para cada questão eliminando os artigos duplicados.

Tabela 21 – Total de artigos retornados eliminando as duplicatas

Questão – Características Relacionadas	**# Artigos Retornados sem Duplicatas**	
	Scopus (até 2010)	**EI Compendex (2007-2010)**
Q1 – Tamanho e Complexidade	265	19
Q2 – Tamanho e Confiabilidade	90	6
Q3 – Complexidade e Esforço	104	13
Q4 – Confiabilidade e Esforço	59	5
Q5 – Complexidade e Manutenibilidade	38	3
Q6 – Esforço e Manutenibilidade	37	11
Q7 – Periodicidade e Esforço	0	1
Q8 – Periodicidade e Manutenibilidade	0	0
Q9 – Tamanho e Esforço	155	11
Q10 – Tamanho e Manutenibilidade	21	2
Q11 – Periodicidade e Tamanho	0	1
Q12 – Complexidade e Confiabilidade	17	5
Q13 – Periodicidade e Complexidade	0	2
Q14 – Manutenibilidade e Confiabilidade	17	2
Q15 – Periodicidade e Confiabilidade	0	0
Total de Artigos	**803**	**81**

Todos os artigos retornados da execução das *strings* de busca passaram por uma primeira análise onde o pesquisador fez uma seleção, com base nos critérios de seleção, através da leitura do título e do *abstract* dos artigos. Após essa primeira análise, foram selecionados 78 artigos retornados da Scopus e 19 retornados da EI Compendex para serem analisados mais detalhadamente. A tabela 22 mostra a quantidade de artigos por questão após a primeira análise.

Tabela 22 – Quantidade de artigos por questão após primeira análise

Questão – Características Relacionadas	**# Artigos após primeira análise**	
	Scopus (até 2010)	**EI Compendex (2007-2010)**
Q1 – Tamanho e Complexidade	30	3
Q2 – Tamanho e Confiabilidade	8	3
Q3 – Complexidade e Esforço	7	3
Q4 – Confiabilidade e Esforço	1	1

Q5 – Complexidade e Manutenibilidade	5	0
Q6 – Esforço e Manutenibilidade	4	1
Q7 – Periodicidade e Esforço	0	0
Q8 – Periodicidade e Manutenibilidade	0	0
Q9 – Tamanho e Esforço	17	7
Q10 – Tamanho e Manutenibilidade	1	0
Q11 – Periodicidade e Tamanho	0	0
Q12 – Complexidade e Confiabilidade	6	1
Q13 – Periodicidade e Complexidade	0	0
Q14 – Manutenibilidade e Confiabilidade	0	0
Q15 – Periodicidade e Confiabilidade	0	0
Total de Artigos	**79**	**19**

A análise mais detalhada dos artigos consiste em uma leitura completa do artigo visando os critérios de inclusão e exclusão definidos no protocolo de *quasi* revisão sistemática. Após a leitura completa dos 79 artigos retornados pela máquina de busca *Scopus*, foram selecionados 24 artigos, e após a leitura completa dos 19 artigos retornados pela máquina de busca *EI Compendex*, foram selecionados 7 artigos, dentre os 7 artigos selecionados da *EI Compendex*. Novamente a seleção dos artigos foi baseada nos critérios de inclusão previamente definidos. Entre os 24 artigos selecionados da Scopus e os 7 artigos selecionados da EI Compendex, existiam 4 artigos em comum.

Após a análise detalhada dos artigos foi realizada a extração de informações dos artigos apresentada no Apêndice A.

Portanto, um total de 27 artigos, já que 4 eram duplicados entre as máquinas de busca, foram selecionados por fornecerem evidências para as questões de pesquisa definidas no protocolo. A tabela 23 mostra a quantidade de artigos por questão. A seção seguinte irá apresentar uma analise da qualidade dos artigos de acordo com os padrões para a apresentação dos resultados da revisão apresentadas no protocolo.

Tabela 23 – Quantidade de artigos por questão após análise detalhada

Questão – Características Relacionadas	**# Artigos após análise detalhada**	
	Scopus (até 2010)	**EI Compendex (2007-2010)**
Q1 – Tamanho e Complexidade	**1**	**1**
Q2 – Tamanho e Confiabilidade	**5**	0
Q3 – Complexidade e Esforço	**5**	**2**
Q4 – Confiabilidade e Esforço	0	0
Q5 – Complexidade e Manutenibilidade	**2**	0
Q6 – Esforço e Manutenibilidade	**1**	0
Q7 – Periodicidade e Esforço	0	0
Q8 – Periodicidade e Manutenibilidade	0	0
Q9 – Tamanho e Esforço	**5**	**3**
Q10 – Tamanho e Manutenibilidade	**1**	0

Q11 – Periodicidade e Tamanho	**1**	0
Q12 – Complexidade e Confiabilidade	**3**	**1**
Q13 – Periodicidade e Complexidade	0	0
Q14 – Manutenibilidade e Confiabilidade	0	0
Q15 – Periodicidade e Confiabilidade	0	0
Total de Artigos	**24**	**7**

4.1. Qualidade dos artigos selecionados

Nesta seção serão apresentados os 31 artigos selecionados e para cada um será apresentada uma análise da qualidade dos artigos de acordo com o padrão definido do protocolo.

Questão 01: Avaliação da influência entre Tamanho e Complexidade

Para a questão P1 (influência entre as características de software Tamanho e Complexidade):

Artigo	Sim	Não	Não Conclusivo
K.K.Aggarwal, Yogesh Singh, Arvinder Kaur, Ruchika Malhotra: **"Empirical Study of Object-Oriented Metrics"**, in *Journal of Object Technology*, vol. 5. no. 8, Novmeber-December 2006, pp. 149-173			X
W. Heijstek; M.R.V. Chaudro; **Empirical Investigations of Model Size, Complexity and Effort in a Large Scale, Distributed Model Driven Development Process.** Software Engineering and Advanced Applications, 2009. SEAA '09. 35th Euromicro Conference on 2009 , Page(s): 113 – 120	X		

Para a questão P1.1 (direção da influência entre as características de software Tamanho e Complexidade):

Artigo	A →B	B →A	Não Conclusivo
K.K.Aggarwal, Yogesh Singh, Arvinder Kaur, Ruchika Malhotra: **"Empirical Study of Object-Oriented Metrics"**, in *Journal of Object Technology*, vol. 5. no. 8, Novmeber-December 2006, pp. 149-173			X
W. Heijstek; M.R.V. Chaudro; **Empirical Investigations of Model Size, Complexity and Effort in a Large Scale, Distributed Model Driven Development Process.** Software Engineering and Advanced Applications, 2009. SEAA '09. 35th Euromicro Conference on 2009 , Page(s): 113 – 120	X		

Para a questão P1.1.1 (intensidade/taxa da influência entre as características de software Tamanho e Complexidade):

Artigo	Taxa	Não Conclusivo
K.K.Aggarwal, Yogesh Singh, Arvinder Kaur, Ruchika Malhotra: **"Empirical Study of Object-Oriented Metrics"**, in *Journal of Object Technology*, vol. 5. no. 8, Novmeber-December 2006, pp. 149-173		X
W. Heijstek; M.R.V. Chaudro; **Empirical**		X

Investigations of Model Size, Complexity and Effort in a Large Scale, Distributed Model Driven Development Process. Software Engineering and Advanced Applications, 2009. SEAA '09. 35th Euromicro Conference on 2009 , Page(s): 113 – 120		

A tabela 24 apresenta uma avaliação sobre a qualidade dos artigos relacionados à questão 1.

Tabela 24 - Avaliação da qualidade dos artigos selecionados para Q1

Questão	**Pontuação de Qualidade por Artigo**					
	Artigo 1	**Artigo 2**				
1. Os dados foram analisados apropriadamente?	NÃO	NÃO				
1.1. Os dados foram analisados para identificar *outliers* e para avaliar as propriedades da distribuição antes da análise?	0	0				
1.2. Os resultados da análise foram utilizados apropriadamente para transformar os dados e para selecionar os *data points* apropriados?	0	0				
2. O estudo realizou análise de sensibilidade ou residual?	NÃO	NÃO				
2.1 Os modelos de estimativa sofreram análise de sensibilidade ou residual?	0	0				
2.2 Os resultados da análise de sensibilidade ou residual foram utilizados para remover dados anormais quando necessário?	0	0				
3. Foram utilizados métodos estatísticos adequados com base na escala de dados brutos e o tipo de distribuição?	1	1				
4. Quão bons foram os métodos de comparação utilizados no estudo?	0.33	0				
5. Estão definidos ou indicados quais foram os projetos utilizados para construir cada modelo?	1	1				
6. Está descrito ou sugerido como a precisão do modelo (erro estimativa) foi medida?	1	1				
7. Está explicito qual o método de validação cruzada foi utilizado?	0	0				
8. Todos os métodos do constructo (modelo) foram totalmente definidos (ferramentas e métodos usados)?	0	1				
Total da pontuação dos artigos primários	4.33	5				

Questão 02: Avaliação da influência entre Tamanho e Confiabilidade

Para a questão P1 (influência entre as características de software Tamanho e Confiabilidade):

Artigo	Sim	Não	Não Conclusivo
El Emam, K. Benlarbi, S. Goel, N. Melo, W. Lounis, H. Rai, S.N. **The optimal class size for object-oriented software**. *Software Engineering, IEEE Transactions on* page(s): 494 - 509 , Volume: 28 Issue: 5, May 2002	X		
Zhang H., **"An investigation of the relationships between lines of code and defects,"** icsm, pp.274-283, 2009 IEEE International Conference on Software Maintenance, 2009	X		
M. English , C. Exton , I. Rigon , B. Cleary, **Fault detection and prediction in an open-source software project**. *Proceedings of the 5th International Conference on Predictor Models in Software Engineering, May 18-19, 2009, Vancouver, British Columbia, Canada*	X		
Leszak, M. **Software Defect Analysis of a Multi-release Telecommunications System** *In Product Focused Software Process Improvement* (2005), pp. 98-114.		X	
N.E. Fenton; M. Neil, W. Marsh, P. Hearty, L. Radlinski, P. Krause. **On the effectiveness of early life cycle defect prediction with Bayesian Nets**. In Proceedings of Empirical Software Engineering. 2008, 499-537.	X		
W. Heijstek; M.R.V. Chaudro; **Empirical Investigations of Model Size, Complexity and Effort in a Large Scale, Distributed Model Driven Development Process.** Software Engineering and Advanced Applications, 2009. SEAA '09. 35th Euromicro Conference on 2009 , Page(s): 113 – 120	X		

Para a questão P1.1 (direção da influência entre as características de software Tamanho e Confiabilidade):

Artigo	A →B	B →A	Não Conclusivo
El Emam, K. Benlarbi, S. Goel, N. Melo, W. Lounis, H. Rai, S.N. **The optimal class size for object-oriented software**. *Software Engineering, IEEE Transactions on* page(s): 494 - 509 , Volume: 28 Issue: 5, May 2002	X		
Hongyu Zhang, **"An investigation of the relationships between lines of code and defects,"** icsm, pp.274-283, 2009 IEEE International Conference on Software Maintenance, 2009	X		
M. English , C. Exton , I. Rigon , B. Cleary, **Fault detection and prediction in an open-source software project**. *Proceedings of the 5th International Conference on Predictor Models in Software Engineering, May 18-19, 2009, Vancouver, British Columbia, Canada*	X		
Leszak, M. **Software Defect Analysis of a Multi-release Telecommunications System** *In Product Focused Software Process Improvement* (2005), pp. 98-114.			X

N.E. Fenton; M. Neil, W. Marsh, P. Hearty, L. Radlinski, P. Krause. **On the effectiveness of early life cycle defect prediction with Bayesian Nets**. In Proceedings of Empirical Software Engineering. 2008, 499-537.	X		
W. Heijstek; M.R.V. Chaudro; **Empirical Investigations of Model Size, Complexity and Effort in a Large Scale, Distributed Model Driven Development Process.** Software Engineering and Advanced Applications, 2009. SEAA '09. 35th Euromicro Conference on 2009 , Page(s): 113 – 120	X		

Para a questão P1.1.1 (intensidade/taxa da influência entre as características de software Tamanho e Confiabilidade):

Artigo	Taxa	Não Conclusivo
El Emam, K. Benlarbi, S. Goel, N. Melo, W. Lounis, H. Rai, S.N. **The optimal class size for object-oriented software**. *Software Engineering, IEEE Transactions on* page(s): 494 - 509 , Volume: 28 Issue: 5, May 2002		X
Hongyu Zhang, **"An investigation of the relationships between lines of code and defects,"** icsm, pp.274-283, 2009 IEEE International Conference on Software Maintenance, 2009	X	
M. English , C. Exton , I. Rigon , B. Cleary, **Fault detection and prediction in an open-source software project**. *Proceedings of the 5th International Conference on Predictor Models in Software Engineering, May 18-19, 2009, Vancouver, British Columbia, Canada*		X
Leszak, M. **Software Defect Analysis of a Multi-release Telecommunications System** *In Product Focused Software Process Improvement* (2005), pp. 98-114.		X
N.E. Fenton; M. Neil, W. Marsh, P. Hearty, L. Radlinski, P. Krause. **On the effectiveness of early life cycle defect prediction with Bayesian Nets**. In Proceedings of Empirical Software Engineering. 2008, 499-537.		X
W. Heijstek; M.R.V. Chaudro; **Empirical Investigations of Model Size, Complexity and Effort in a Large Scale, Distributed Model Driven Development Process.** Software Engineering and Advanced Applications, 2009. SEAA '09. 35th Euromicro Conference on 2009 , Page(s): 113 – 120		X

A tabela 25 apresenta uma avaliação sobre a qualidade dos artigos relacionados à questão 2.

Tabela 25- Avaliação da qualidade dos artigos selecionados para Q2

Questão	Pontuação de Qualidade por Artigo					
	Artigo 1	Artigo 2	Artigo 3	Artigo 4	Artigo 5	Artigo 6
1. Os dados foram analisados apropriadamente?	SIM	NÃO	NÃO	NÃO	NÃO	NÃO

1.1. Os dados foram analisados para identificar *outliers* e para avaliar as propriedades da distribuição antes da análise?	0.5	0	0	0	0	0
1.2. Os resultados da análise foram utilizados apropriadamente para transformar os dados e para selecionar os *data points* apropriados?	0.5	0	0	0	0	0
2. O estudo realizou análise de sensibilidade ou residual?	NÃO	NÃO	NÃO	NÃO	SIM	NÃO
2.1 Os modelos de estimativa sofreram análise de sensibilidade ou residual?	0	0	0	0	0.5	0
2.2 Os resultados da análise de sensibilidade ou residual foram utilizados para remover dados anormais quando necessário?	0	0	0	0	0.5	0
3. Foram utilizados métodos estatísticos adequados com base na escala de dados brutos e o tipo de distribuição?	1	1	1	1	1	1
4. Quão bons foram os métodos de comparação utilizados no estudo?	0.33	1	0	0	1	0
5. Estão definidos ou indicados quais foram os projetos utilizados para construir cada modelo?	1	1	1	1	0	1
6. Está descrito ou sugerido como a precisão do modelo (erro estimativa) foi medida?	1	1	1	0	1	1
7. Está explicito qual o método de validação cruzada foi utilizado?	0	1	0	0	0	0
8. Todos os métodos do constructo (modelo) foram totalmente definidos (ferramentas e métodos usados)?	0	1	0	1	1	1
Total da pontuação dos artigos primários	6.33	7	4	4	7	5

Questão 03: Avaliação da influência entre Complexidade e Esforço

Para a questão P1 (influência entre as características de software Complexidade e Esforço):

Artigo	Sim	Não	Não Conclusivo
L. Yu , S. Ramaswamy, **Component dependency in object-oriented software**, *Journal of Computer Science and Technology*, v.22 n.3, p.379-386, May 2007	X		
D.P. Darcy, C.F. Kemerer, S.A. Slaughter and J.E. Tomayko**, The structural complexity of software: An experimental test**, IEEE Trans. Softw. Eng. 31 (11) (2005), pp. 982–995	X		
M. G. Bocco , D. L. Moody , M. Piattini, **Assessing the capability of internal metrics as early indicators of maintenance effort through experimentation: Research Articles,** *Journal of Software Maintenance and Evolution: Research and Practice*, v.17 n.3, p.225-246, May 2005	X		
Kozlov D, Koskinen J, Sakkinen M, Markkula J. **Assessing maintainability change over multiple software releases**. *Journal of Software Maintenance and Evolution: Research and Practice* 2008; 20:31–58.	X		
SENTAS P.; L. ANGELIS; I. STAMELOS (2008). **A Statistical Framework for Analyzing the Duration of Software Projects**. *Empirical Software Engineering (Springer)*. 13:147–184.	X		

Para a questão P1.1 (direção da influência entre as características de software Complexidade e Esforço):

Artigo	A →B	B →A	Não Conclusivo
L. Yu , S. Ramaswamy, **Component dependency in object-oriented software**, *Journal of Computer Science and Technology*, v.22 n.3, p.379-386, May 2007	X		
D.P. Darcy, C.F. Kemerer, S.A. Slaughter and J.E. Tomayko**, The structural complexity of software: An experimental test**, IEEE Trans. Softw. Eng. 31 (11) (2005), pp. 982–995	X		
M. G. Bocco , D. L. Moody , M. Piattini, **Assessing the capability of internal metrics as early indicators of maintenance effort through experimentation: Research Articles,** *Journal of Software Maintenance and Evolution: Research and Practice*, v.17 n.3, p.225-246, May 2005	X		
Kozlov D, Koskinen J, Sakkinen M, Markkula J. **Assessing maintainability change over multiple software releases**. *Journal of Software Maintenance and Evolution: Research and Practice* 2008; 20:31–58.	X		
SENTAS P.; L. ANGELIS; I. STAMELOS (2008). **A Statistical Framework for Analyzing the Duration of Software Projects**. *Empirical Software Engineering (Springer)*. 13:147–184.	X		

Para a questão P1.1.1 (intensidade/taxa da influência entre as características de software Complexidade e Esforço):

Artigo	Taxa	Não Conclusivo
L. Yu , S. Ramaswamy, **Component dependency in object-oriented software**, *Journal of Computer Science and Technology*, v.22 n.3, p.379-386, May 2007	X	
D.P. Darcy, C.F. Kemerer, S.A. Slaughter and J.E. Tomayko**, The structural complexity of software: An experimental test**, IEEE Trans. Softw. Eng. 31 (11) (2005), pp. 982–995	X	
M. G. Bocco , D. L. Moody , M. Piattini, **Assessing the capability of internal metrics as early indicators of maintenance effort through experimentation: Research Articles,** *Journal of Software Maintenance and Evolution: Research and Practice*, v.17 n.3, p.225-246, May 2005	X	
Kozlov D, Koskinen J, Sakkinen M, Markkula J. **Assessing maintainability change over multiple software releases**. *Journal of Software Maintenance and Evolution: Research and Practice* 2008; 20:31–58.	X	
SENTAS P.; L. ANGELIS; I. STAMELOS (2008). **A Statistical Framework for Analyzing the Duration of Software Projects**. *Empirical Software Engineering (Springer)*. 13:147–184.		X

A tabela 26 apresenta uma avaliação sobre a qualidade dos artigos relacionados à questão 3.

Tabela 26- Avaliação da qualidade dos artigos selecionados para Q3

Questão	Pontuação de Qualidade por Artigo				
	Artigo 1	Artigo 2	Artigo 3	Artigo 4	Artigo 5
1.Os dados foram analisados apropriadamente?	NÃO	SIM	NÃO	NÃO	NÃO
1.1. Os dados foram analisados para identificar *outliers* e para avaliar as propriedades da distribuição antes da análise?	0	0.5	0	0	0
1.2. Os resultados da análise foram utilizados apropriadamente para transformar os dados e para selecionar os *data points* apropriados?	0	0.5	0	0	0
2. O estudo realizou análise de sensibilidade ou residual?	NÃO	NÃO	NÃO	NÃO	SIM
2.1 Os modelos de estimativa sofreram análise de sensibilidade ou residual?	0	0	0	0	1
2.2 Os resultados da análise de sensibilidade ou residual foram utilizados para remover dados anormais quando necessário?	0	0	0	0	0
3. Foram utilizados métodos estatísticos adequados com base na escala de dados brutos e o tipo de distribuição?	1	1	1	1	1
4. Quão bons foram os métodos de comparação utilizados no estudo?	0.33	0.33	0.67	0.33	0.67

5. Estão definidos ou indicados quais foram os projetos utilizados para construir cada modelo?	1	1	1	1	0
6. Está descrito ou sugerido como a precisão do modelo (erro estimativa) foi medida?	0	1	1	1	1
7. Está explicito qual o método de validação cruzada foi utilizado?	0	0	0	0	0
8. Todos os métodos do constructo (modelo) foram totalmente definidos (ferramentas e métodos usados)?	0	1	1	1	1
Total da pontuação dos artigos primários	3.33	7.33	5.67	5.33	5.67

Questão 04: Avaliação da influência entre Esforço e Confiabilidade

Nenhuma evidência adicional foi encontrada para esta questão.

Questão 05: Avaliação da influência entre Complexidade e Manutenibilidade

Para a questão P1 (influência entre as características de software Complexidade e Manutenibilidade):

Artigo	Sim	Não	Não Conclusivo
Kozlov D, Koskinen J, Sakkinen M, Markkula J. **Assessing maintainability change over multiple software releases**. *Journal of Software Maintenance and Evolution: Research and Practice* 2008; 20:31–58.	X		
J. A. Cruz-Lemus, M. Genero, M. Piattini,**Using Controlled Experiments for Validating UML Statechart Diagrams Measures** *Software Process and Product Measurement. ACM* 2007	X		
G. Poels , G. Dedene, **Evaluating the Effect of Inheritance on the Modifiability of Object-Oriented Business Domain Models**, *Proceedings of the Fifth European Conference on Software Maintenance and Reengineering,* p.20, March 14-16, 2001	X		
G. Canfora , F. García , M. Piattini , F. Ruiz , C. A. Visaggio, **A family of experiments to validate metrics for software process models***, Journal of Systems and Software, v.77* n.2, p.113-129, August 2005	X		
Riaz, M.; Mendes, E.; Tempero, E. A. 2009. **Systematic Review of Software Maintainability Prediction and Metrics.** Proceeding ESEM '09 Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement	X		

Para a questão P1.1 (direção da influência entre as características de software Complexidade e Manutenibilidade):

Artigo	A →B	B →A	Não Conclusivo
Kozlov D, Koskinen J, Sakkinen M, Markkula J. **Assessing maintainability change over multiple software releases**. *Journal of Software Maintenance and Evolution: Research and Practice* 2008; 20:31–58.	X		

J. A. Cruz-Lemus, M. Genero, M. Piattini,**Using Controlled Experiments for Validating UML Statechart Diagrams Measures** *Software Process and Product Measurement. ACM* 2007	X		
G. Poels , G. Dedene, **Evaluating the Effect of Inheritance on the Modifiability of Object-Oriented Business Domain Models**, *Proceedings of the Fifth European Conference on Software Maintenance and Reengineering,* p.20, March 14-16, 2001	X		
G. Canfora , F. García , M. Piattini , F. Ruiz , C. A. Visaggio, **A family of experiments to validate metrics for software process models***, Journal of Systems and Software, v.77* n.2, p.113-129, August 2005	X		
Riaz, M.; Mendes, E.; Tempero, E. A. 2009. **Systematic Review of Software Maintainability Prediction and Metrics.** Proceeding ESEM '09 Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement	X		

Para a questão P1.1.1 (intensidade/taxa da influência entre as características de software Complexidade e Manutenibilidade):

Artigo	**Taxa**	**Não Conclusivo**
Kozlov D, Koskinen J, Sakkinen M, Markkula J. **Assessing maintainability change over multiple software releases**. *Journal of Software Maintenance and Evolution: Research and Practice* 2008; 20:31–58.	X	
J. A. Cruz-Lemus, M. Genero, M. Piattini,**Using Controlled Experiments for Validating UML Statechart Diagrams Measures** *Software Process and Product Measurement. ACM* 2007	X	
G. Poels , G. Dedene, **Evaluating the Effect of Inheritance on the Modifiability of Object-Oriented Business Domain Models**, *Proceedings of the Fifth European Conference on Software Maintenance and Reengineering,* p.20, March 14-16, 2001		X
G. Canfora , F. García , M. Piattini , F. Ruiz , C. A. Visaggio, **A family of experiments to validate metrics for software process models***, Journal of Systems and Software, v.77* n.2, p.113-129, August 2005		X
Riaz, M.; Mendes, E.; Tempero, E. A. 2009. **Systematic Review of Software Maintainability Prediction and Metrics.** Proceeding ESEM '09 Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement		X

A tabela 27 apresenta uma avaliação sobre a qualidade dos artigos relacionados à questão 5.

Tabela 27- Avaliação da qualidade dos artigos selecionados para Q5

Questão	Pontuação de Qualidade por Artigo				
	Artigo 1	**Artigo 2**	**Artigo 3**	**Artigo 4**	**Artigo 5**
1. Os dados foram analisados apropriadamente?	NÃO	NÃO	NÃO	NÃO	NÃO
1.1. Os dados foram analisados para identificar *outliers* e para avaliar as propriedades da distribuição antes da análise?	0	0	0	0	0
1.2. Os resultados da análise foram utilizados apropriadamente para transformar os dados e para selecionar os *data points* apropriados?	0	0	0	0	0
2. O estudo realizou análise de sensibilidade ou residual?	NÃO	NÃO	NÃO	NÃO	NÃO
2.1 Os modelos de estimativa sofreram análise de sensibilidade ou residual?	0	0	0	0	0
2.2 Os resultados da análise de sensibilidade ou residual foram utilizados para remover dados anormais quando necessário?	0	0	0	0	0
3. Foram utilizados métodos estatísticos adequados com base na escala de dados brutos e o tipo de distribuição?	1	1	1	1	0
4. Quão bons foram os métodos de comparação utilizados no estudo?	0.33	0.67	0.33	1	1
5. Estão definidos ou indicados quais foram os projetos utilizados para construir cada modelo?	1	0	1	0	1
6. Está descrito ou sugerido como a precisão do modelo (erro estimativa) foi medida?	1	1	1	1	1
7. Está explicito qual o método de validação cruzada foi utilizado?	0	0	0	0	0
8. Todos os métodos do constructo (modelo) foram totalmente definidos (ferramentas e métodos usados)?	1	1	1	0	1
Total da pontuação dos artigos primários	5.33	4.67	5.33	4	4

Questão 06: Avaliação da influência entre Esforço e Manutenibilidade

Para a questão P1 (influência entre as características de software Esforço e Manutenibilidade):

Artigo	Sim	Não	Não Conclusivo
Riaz, M.; Mendes, E.; Tempero, E. A. 2009. **Systematic Review of Software Maintainability Prediction and Metrics.** Proceeding ESEM '09 Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement	X		

Para a questão P1.1 (direção da influência entre as características de software Complexidade e Manutenibilidade):

Artigo	A →B	B →A	Não Conclusivo
Riaz, M.; Mendes, E.; Tempero, E. A. 2009. **Systematic Review of Software Maintainability Prediction and Metrics.** Proceeding ESEM '09 Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement	X		

Para a questão P1.1.1 (intensidade/taxa da influência entre as características de software Complexidade e Manutenibilidade):

Artigo	Taxa	Não Conclusivo
Riaz, M.; Mendes, E.; Tempero, E. A. 2009. **Systematic Review of Software Maintainability Prediction and Metrics.** Proceeding ESEM '09 Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement		X

A tabela 28 apresenta uma avaliação sobre a qualidade dos artigos relacionados à questão 6.

Tabela 28- Avaliação da qualidade dos artigos selecionados para Q6

Questão	Pontuação de Qualidade por Artigo				
	Artigo 1				
1. Os dados foram analisados apropriadamente?	NÃO				
1.1. Os dados foram analisados para identificar *outliers* e para avaliar as propriedades da distribuição antes da análise?	0				
1.2. Os resultados da análise foram utilizados apropriadamente para transformar os dados e para selecionar os *data points* apropriados?	0				
2. O estudo realizou análise de sensibilidade ou residual?	NÃO				
2.1 Os modelos de estimativa sofreram análise de sensibilidade ou residual?	0				
2.2 Os resultados da análise de sensibilidade ou residual foram utilizados para remover dados anormais quando necessário?	0				
3. Foram utilizados métodos estatísticos adequados com base na	0				

escala de dados brutos e o tipo de distribuição?					
4. Quão bons foram os métodos de comparação utilizados no estudo?	1				
5. Estão definidos ou indicados quais foram os projetos utilizados para construir cada modelo?	1				
6. Está descrito ou sugerido como a precisão do modelo (erro estimativa) foi medida?	1				
7. Está explicito qual o método de validação cruzada foi utilizado?	0				
8. Todos os métodos do constructo (modelo) foram totalmente definidos (ferramentas e métodos usados)?	1				
Total da pontuação dos artigos primários	4				

Questão 07: Avaliação da influência entre Esforço e Periodicidade

Nenhuma evidência adicional foi encontrada para esta questão.

Questão 08: Avaliação da influência entre Periodicidade e Manutenibilidade

Nenhuma evidência adicional foi encontrada para esta questão. Portanto este relacionamento continua como não existente.

Questão 09: Avaliação da influência entre Tamanho e Esforço

Para a questão P1 (influência entre as características de software Tamanho e Esforço):

Artigo	Sim	Não	Não Conclusivo
Kozlov D, Koskinen J, Sakkinen M, Markkula J. **Assessing maintainability change over multiple software releases**. *Journal of Software Maintenance and Evolution: Research and Practice* 2008; 20:31–58.	X		
SENTAS P.; L. ANGELIS; I. STAMELOS (2008). **A Statistical Framework for Analyzing the Duration of Software Projects**. *Empirical Software Engineering (Springer)*. 13:147–184.	X		
J. J. Dolado, **A study of the relationships among Albrecht and Mark II function points, lines of code 4GL and effort**, *Journal of Systems and Software*, v.37 n.2, p.161-173, May 1997	X		
F. Ferrucci, C. Gravino, S. Di Martino, **A case study using Web objects and COSMIC for effort estimation of Web applications**, *in: Proceedings of the 34th Euromicro Conference on Software Engineering and Advanced Applications (SEAA'08), Parma, Italy, 2008, pp. 441-448.*	X		
J. Zhang, **"The Establishment and Application of Effort Regression Equation,"** csse, vol. 2, pp.11-14, 2008 *International Conference on Computer Science and Software Engineering, 2008b*	X		
Yinhuan Zheng; B. Wang; Yilong Zheng; L. Shi; **Estimation of software projects effort based on function point.** *Computer Science & Education, 2009. ICCSE '09. 4th International Conference on. pp.* 941 - 943	X		
J.J. Dolado**, On the Problem of the Software Cost Function**, *Information Software Technology*, vol. 43, no. 1, pp. 61-72, 2001.		X	
M. Tsunoda , A. Monden , H. Yadohisa , N. Kikuchi , K. Matsumoto, **Software development productivity of Japanese enterprise applications**, *Information Technology and Management*, v.10 n.4, p.193-205, *December* 2009		X	
W. Heijstek; M.R.V. Chaudro; **Empirical Investigations of Model Size, Complexity and Effort in a Large Scale, Distributed Model**	X		

Driven Development Process. Software Engineering and Advanced Applications, 2009. SEAA '09. 35th Euromicro Conference on 2009 , Page(s): 113 – 120			
Zhang Jun-guang; **Method Study of Software Project Effort Estimation**. Wireless Communications, Networking and Mobile Computing, 2008.WiCOM '08.4th International Conference on.2008a. p:1-4	X		

Para a questão P1.1 (direção da influência entre as características de software Tamanho e Esforço):

Artigo	A →B	B →A	Não Conclusivo
Kozlov D, Koskinen J, Sakkinen M, Markkula J. **Assessing maintainability change over multiple software releases**. *Journal of Software Maintenance and Evolution: Research and Practice* 2008; 20:31–58.	X		
SENTAS P.; L. ANGELIS; I. STAMELOS (2008). **A Statistical Framework for Analyzing the Duration of Software Projects**. *Empirical Software Engineering (Springer)*. 13:147–184.	X		
J. J. Dolado, **A study of the relationships among Albrecht and Mark II function points, lines of code 4GL and effort**, *Journal of Systems and Software*, v.37 n.2, p.161-173, May 1997	X		
F. Ferrucci, C. Gravino, S. Di Martino, **A case study using Web objects and COSMIC for effort estimation of Web applications**, *in: Proceedings of the 34th Euromicro Conference on Software Engineering and Advanced Applications (SEAA'08), Parma, Italy, 2008, pp. 441-448.*			X
J. Zhang, **"The Establishment and Application of Effort Regression Equation,"** csse, vol. 2, pp.11-14, 2008 *International Conference on Computer Science and Software Engineering, 2008*	X		
Yinhuan Zheng; B. Wang; Yilong Zheng; L. Shi; **Estimation of software projects effort based on function point.** *Computer Science & Education, 2009. ICCSE '09. 4th International Conference on. pp*. 941 - 943	X		
J.J. Dolado**, On the Problem of the Software Cost Function**, *Information Software Technology*, vol. 43, no. 1, pp. 61-72, 2001.			
M. Tsunoda , A. Monden , H. Yadohisa , N. Kikuchi , K. Matsumoto, **Software development productivity of Japanese enterprise applications**, *Information Technology and Management*, v.10 n.4, p.193-205, *December* 2009			X
W. Heijstek; M.R.V. Chaudro; **Empirical Investigations of Model Size, Complexity and Effort in a Large Scale, Distributed Model Driven Development Process.** Software Engineering and Advanced Applications, 2009. SEAA '09. 35th Euromicro Conference on 2009 , Page(s): 113 – 120	X		
Zhang Jun-guang; **Method Study of Software Project Effort Estimation**. Wireless Communications, Networking and Mobile Computing, 2008.WiCOM '08.4th International	X		

Conference on.2008.pages:1-4			

Para a questão P1.1.1 (intensidade/taxa da influência entre as características de software Tamanho e Esforço):

Artigo	Taxa	Não Conclusivo
Kozlov D, Koskinen J, Sakkinen M, Markkula J. **Assessing maintainability change over multiple software releases**. *Journal of Software Maintenance and Evolution: Research and Practice* 2008; 20:31–58.		X
SENTAS P.; L. ANGELIS; I. STAMELOS (2008). **A Statistical Framework for Analyzing the Duration of Software Projects**. *Empirical Software Engineering (Springer)*. 13:147–184.		X
J. J. Dolado, **A study of the relationships among Albrecht and Mark II function points, lines of code 4GL and effort**, *Journal of Systems and Software*, v.37 n.2, p.161-173, May 1997		X
F. Ferrucci, C. Gravino, S. Di Martino, **A case study using Web objects and COSMIC for effort estimation of Web applications**, *in: Proceedings of the 34th Euromicro Conference on Software Engineering and Advanced Applications (SEAA'08), Parma, Italy, 2008, pp. 441-448.*		X
J. Zhang, **"The Establishment and Application of Effort Regression Equation,"** csse, vol. 2, pp.11-14, 2008 *International Conference on Computer Science and Software Engineering, 2008*		X
Yinhuan Zheng; B. Wang; Yilong Zheng; L. Shi; **Estimation of software projects effort based on function point.** *Computer Science & Education, 2009. ICCSE '09. 4th International Conference on. pp*. 941 - 943	X	
J.J. Dolado**, On the Problem of the Software Cost Function**, *Information Software Technology*, vol. 43, no. 1, pp. 61-72, 2001.		X
M. Tsunoda , A. Monden , H. Yadohisa , N. Kikuchi , K. Matsumoto, **Software development productivity of Japanese enterprise applications**, *Information Technology and Management*, v.10 n.4, p.193-205, *December* 2009		X
W. Heijstek; M.R.V. Chaudro; **Empirical Investigations of Model Size, Complexity and Effort in a Large Scale, Distributed Model Driven Development Process.** Software Engineering and Advanced Applications, 2009. SEAA '09. 35th Euromicro Conference on 2009 , Page(s): 113 – 120		X
Zhang Jun-guang; **Method Study of Software Project Effort Estimation**. Wireless Communications, Networking and Mobile Computing, 2008.WiCOM '08.4th International Conference on.2008.pages:1-4		X

A tabela 29 apresenta uma avaliação sobre a qualidade dos artigos relacionados à questão 9.

Tabela 29- Avaliação da qualidade dos artigos selecionados para Q9

Questão	Pontuação de Qualidade por Artigo									
	Artigo 1	**Artigo 2**	**Artigo 3**	**Artigo 4**	**Artigo 5**	**Artigo 6**	**Artigo 7**	**Artigo 8**	**Artigo 9**	**Artigo10**
1. Os dados foram analisados apropriadamente?	NÃO	NÃO	NÃO	SIM	NÃO	NÃO	SIM	SIM	NÃO	NÃO
1.1. Os dados foram analisados para identificar *outliers* e para avaliar as propriedades da distribuição antes da análise?	0	0	0	0.5	0	0	0.5	0.5	0	0
1.2. Os resultados da análise foram utilizados apropriadamente para transformar os dados e para selecionar os *data points* apropriados?	0	0	0	0.5	0	0	0.5	0.5	0	0
2. O estudo realizou análise de sensibilidade ou residual?	NÃO	SIM	NÃO	NÃO	SIM	NÃO	NÃO	NÃO	NÃO	NÃO
2.1 Os modelos de estimativa sofreram análise de sensibilidade ou residual?	0	1	0	0	0.5	0	0	0	0	0
2.2 Os resultados da análise de sensibilidade ou residual foram utilizados para remover dados anormais quando necessário?	0	0	0	0	0.5	0	0	0	0	0
3. Foram utilizados métodos estatísticos adequados com base na escala de dados brutos e o tipo de distribuição?	1	1	1	1	1	0	1	1	1	1
4. Quão bons foram os métodos de comparação utilizados no estudo?	0.33	0.67	1	1	1	0.67	0.33	1	0	1

5. Estão definidos ou indicados quais foram os projetos utilizados para construir cada modelo?	1	0	0	0	0	0	0	0	1	0
6. Está descrito ou sugerido como a precisão do modelo (erro estimativa) foi medida?	1	1	0	0	0	0	0	0	1	0
7. Está explicito qual o método de validação cruzada foi utilizado?	0	0	0	0	0	0	0	0	0	0
8. Todos os métodos do constructo (modelo) foram totalmente definidos (ferramentas e métodos usados)?	1	1	0	1	0	0	0	0	1	0
Total da pontuação dos artigos primários	5.33	5.67	3	6	4	0.67	4.33	5	5	3

Questão 10: Avaliação da influência entre Tamanho e Manutenibilidade

Para a questão P1 (influência entre as características de software Tamanho e Manutenibilidade):

Artigo	Sim	Não	Não Conclusivo
Kozlov D, Koskinen J, Sakkinen M, Markkula J. **Assessing maintainability change over multiple software releases**. *Journal of Software Maintenance and Evolution: Research and Practice* 2008; 20:31–58.	X		
J. A. Cruz-Lemus, M. Genero, M. Piattini,**Using Controlled Experiments for Validating UML Statechart Diagrams Measures** *Software Process and Product Measurement. ACM* 2007	X		
G. Canfora , F. García , M. Piattini , F. Ruiz , C. A. Visaggio, **A family of experiments to validate metrics for software process models***, Journal of Systems and Software, v.77* n.2, p.113-129, August 2005	X		
Riaz, M.; Mendes, E.; Tempero, E. A. 2009. **Systematic Review of Software Maintainability Prediction and Metrics.** Proceeding ESEM '09 Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement	X		

Para a questão P1.1 (direção da influência entre as características de software Tamanho e Manutenibilidade):

Artigo	A →B	B →A	Não Conclusivo
Kozlov D, Koskinen J, Sakkinen M, Markkula J. **Assessing maintainability change over multiple software releases**. *Journal of Software Maintenance and Evolution: Research and Practice* 2008; 20:31–58.	X		
J. A. Cruz-Lemus, M. Genero, M. Piattini,**Using Controlled Experiments for Validating UML Statechart Diagrams Measures** *Software Process and Product Measurement. ACM* 2007	X		
G. Canfora , F. García , M. Piattini , F. Ruiz , C. A. Visaggio, **A family of experiments to validate metrics for software process models**, *Journal of Systems and Software, v.77* n.2, p.113-129, August 2005	X		
Riaz, M.; Mendes, E.; Tempero, E. A. 2009. **Systematic Review of Software Maintainability Prediction and Metrics.** Proceeding ESEM '09 Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement	X		

Para a questão P1.1.1 (intensidade/taxa da influência entre as características de software Tamanho e Manutenibilidade):

Artigo	Taxa	Não Conclusivo
Kozlov D, Koskinen J, Sakkinen M, Markkula J. **Assessing maintainability change over multiple software releases**. *Journal of Software Maintenance and Evolution: Research and Practice* 2008; 20:31–58.		X
J. A. Cruz-Lemus, M. Genero, M. Piattini,**Using Controlled Experiments for Validating UML Statechart Diagrams Measures** *Software Process and Product Measurement. ACM* 2007	X	
G. Canfora , F. García , M. Piattini , F. Ruiz , C. A. Visaggio, **A family of experiments to validate metrics for software process models**, *Journal of Systems and Software, v.77* n.2, p.113-129, August 2005		X
Riaz, M.; Mendes, E.; Tempero, E. A. 2009. **Systematic Review of Software Maintainability Prediction and Metrics.** Proceeding ESEM '09 Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement		X

A tabela 30 apresenta uma avaliação sobre a qualidade dos artigos relacionados à questão 10.

Tabela 30- Avaliação da qualidade dos artigos selecionados para Q10

Questão	**Pontuação de Qualidade por Artigo**				
	Artigo 1	**Artigo 2**	**Artigo 3**	**Artigo 4**	
1. Os dados foram analisados apropriadamente?	NÃO	NÃO	NÃO	NÃO	
1.1. Os dados foram analisados para identificar *outliers* e para avaliar as propriedades da distribuição antes da análise?	0	0	0	0	
1.2. Os resultados da análise foram utilizados apropriadamente para transformar os dados e para selecionar os *data points* apropriados?	0	0	0	0	
2. O estudo realizou análise de sensibilidade ou residual?	NÃO	NÃO	NÃO	NÃO	
2.1 Os modelos de estimativa sofreram análise de sensibilidade ou residual?	0	0	0	0	
2.2 Os resultados da análise de sensibilidade ou residual foram utilizados para remover dados anormais quando necessário?	0	0	0	0	
3. Foram utilizados métodos estatísticos adequados com base na escala de dados brutos e o tipo de distribuição?	1	1	1	0	
4. Quão bons foram os métodos de comparação utilizados no estudo?	0.33	0.67	1	1	
5. Estão definidos ou indicados quais foram os projetos utilizados para construir cada modelo?	1	0	0	1	
6. Está descrito ou sugerido como a precisão do modelo (erro estimativa) foi medida?	1	1	1	1	
7. Está explicito qual o método de validação cruzada foi utilizado?	0	0	0	0	
8. Todos os métodos do constructo (modelo) foram totalmente definidos (ferramentas e métodos usados)?	1	1	0	1	
Total da pontuação dos artigos primários	5.33	4.67	4	4	

Questão 11: Avaliação da influência entre Periodicidade e Tamanho

Nenhuma evidência adicional foi encontrada para esta questão.

Questão 12: Avaliação da influência entre Complexidade e Confiabilidade

Para a questão P1 (influência entre as características de software Complexidade e Confiabilidade):

Artigo	Sim	Não	Não Conclusivo
E. H. Ferneley, **Design metrics as an aid to software maintenance: an empirical study**, *Journal of Software Maintenance: Research and Practice, v.11 n.1, p.55-72, Jan.-Feb. 1999*	X		
K. K. Aggarwal, Yogesh Singh, Arvinder Kaur, Ruchika Malhotra: **Investigating effect of Design Metrics on Fault Proneness in Object-Oriented Systems**. *Journal of Object Technology 6(10): 127-141 (2007)*	X		
M. English , C. Exton , I. Rigon , B. Cleary, **Fault detection and prediction in an open-source software project**. *Proceedings of the 5th International Conference on Predictor Models in Software Engineering, May 18-19, 2009, Vancouver, British Columbia, Canada*	X		
F. Fioravanti , P. Nesi, **A Study on Fault-Proneness Detection of Object-Oriented Systems**, *Proceedings of the Fifth European Conference on Software Maintenance and Reengineering*, p.121, March 14-16, 2001	X		
W. Heijstek; M.R.V. Chaudro; **Empirical Investigations of Model Size, Complexity and Effort in a Large Scale, Distributed Model Driven Development Process.** Software Engineering and Advanced Applications, 2009. SEAA '09. 35th Euromicro Conference on 2009 , Page(s): 113 – 120	X		
N. Schneidewind; M. Hinchey; **A Complexity Reliability Model.** Software Reliability Engineering, 2009. ISSRE '09. 20th International Symposium on. page(s): 1 – 10	X		

Para a questão P1.1 (direção da influência entre as características de software Complexidade e Confiabilidade):

Artigo	A →B	B →A	Não Conclusivo
E. H. Ferneley, **Design metrics as an aid to software maintenance: an empirical study**, *Journal of Software Maintenance: Research and Practice, v.11 n.1, p.55-72, Jan.-Feb. 1999*	X		
K. K. Aggarwal, Yogesh Singh, Arvinder Kaur, Ruchika Malhotra: **Investigating effect of Design Metrics on Fault Proneness in Object-Oriented Systems**. *Journal of Object Technology 6(10): 127-141 (2007)*	X		
M. English , C. Exton , I. Rigon , B. Cleary, **Fault detection and prediction in an open-source software project**. *Proceedings of the 5th International Conference on Predictor Models in Software Engineering, May 18-19, 2009, Vancouver, British Columbia, Canada*	X		
F. Fioravanti , P. Nesi, **A Study on Fault-Proneness Detection of Object-Oriented Systems**, *Proceedings of the Fifth European*	X		

Conference on Software Maintenance and Reengineering, p.121, March 14-16, 2001			
W. Heijstek; M.R.V. Chaudro; **Empirical Investigations of Model Size, Complexity and Effort in a Large Scale, Distributed Model Driven Development Process.** Software Engineering and Advanced Applications, 2009. SEAA '09. 35th Euromicro Conference on 2009 , Page(s): 113 – 120	X		
N. Schneidewind; M. Hinchey; **A Complexity Reliability Model.** Software Reliability Engineering, 2009. ISSRE '09. 20th International Symposium on. page(s): 1 – 10	X		

Para a questão P1.1.1 (intensidade/taxa da influência entre as características de software Complexidade e Confiabilidade):

Artigo	Taxa	Não Conclusivo
E. H. Ferneley, **Design metrics as an aid to software maintenance: an empirical study**, *Journal of Software Maintenance: Research and Practice, v.11 n.1, p.55-72, Jan.-Feb. 1999*		X
K. K. Aggarwal, Yogesh Singh, Arvinder Kaur, Ruchika Malhotra: **Investigating effect of Design Metrics on Fault Proneness in Object-Oriented Systems**. *Journal of Object Technology 6(10): 127-141 (2007)*	X	
M. English , C. Exton , I. Rigon , B. Cleary, **Fault detection and prediction in an open-source software project**. *Proceedings of the 5th International Conference on Predictor Models in Software Engineering, May 18-19, 2009, Vancouver, British Columbia, Canada*		X
F. Fioravanti , P. Nesi, **A Study on Fault-Proneness Detection of Object-Oriented Systems**, *Proceedings of the Fifth European Conference on Software Maintenance and Reengineering*, p.121, March 14-16, 2001		X
W. Heijstek; M.R.V. Chaudro; **Empirical Investigations of Model Size, Complexity and Effort in a Large Scale, Distributed Model Driven Development Process.** Software Engineering and Advanced Applications, 2009. SEAA '09. 35th Euromicro Conference on 2009 , Page(s): 113 – 120		X
N. Schneidewind; M. Hinchey; **A Complexity Reliability Model.** Software Reliability Engineering, 2009. ISSRE '09. 20th International Symposium on. page(s): 1 – 10		X

A tabela 31 apresenta uma avaliação sobre a qualidade dos artigos relacionados à questão 12.

Tabela 31- Avaliação da qualidade dos artigos selecionados para Q12

Questão	**Pontuação de Qualidade por Artigo**					
	Artigo 1	**Artigo 2**	**Artigo 3**	**Artigo 4**	**Artigo 5**	**Artigo 6**
1. Os dados foram analisados apropriadamente?	SIM	NÃO	NÃO	NÃO	NÃO	NÃO
1.1. Os dados foram analisados para identificar *outliers* e para avaliar as propriedades da distribuição antes da análise?	0.5	0	0	0	0	0
1.2. Os resultados da análise foram utilizados apropriadamente para transformar os dados e para selecionar os *data points* apropriados?	0.5	0	0	0	0	0
2. O estudo realizou análise de sensibilidade ou residual?	NÃO	SIM	NÃO	SIM	NÃO	NÃO
2.1 Os modelos de estimativa sofreram análise de sensibilidade ou residual?	0	1	0	1	0	0
2.2 Os resultados da análise de sensibilidade ou residual foram utilizados para remover dados anormais quando necessário?	0	0	0	0	0	0
3. Foram utilizados métodos estatísticos adequados com base na escala de dados brutos e o tipo de distribuição?	1	1	1	1	1	1
4. Quão bons foram os métodos de comparação utilizados no estudo?	0	0.67	0	0.33	0	0
5. Estão definidos ou indicados quais foram os projetos utilizados para construir cada modelo?	0	0	1	0	1	1
6. Está descrito ou sugerido como a precisão do modelo (erro estimativa) foi medida?	1	1	1	0	1	1
7. Está explicito qual o método de validação cruzada foi utilizado?	0	1	0	0	0	0
8. Todos os métodos do constructo (modelo) foram totalmente definidos (ferramentas e métodos usados)?	0	0	0	0	1	1
Total da pontuação dos artigos primários	5	5.67	4	3.33	5	5

Questão 13: Avaliação da influência entre Periodicidade e Complexidade

Nenhuma evidência adicional foi encontrada para esta questão.

Questão 14: Avaliação da influência entre Manutenibilidade e Confiabilidade

Nenhuma evidência adicional foi encontrada para esta questão.

Questão 15: Avaliação da influência entre Periodicidade e Confiabilidade

Nenhuma evidência adicional foi encontrada para esta questão.