Observando Evolução de Software através de Modelos Dinâmicos

Marco Antônio Pereira Araújo, Guilherme Horta Travassos

UFRJ - Universidade Federal do Rio de Janeiro
COPPE – Programa de Engenharia de Sistemas e Computação

maraujo@acessa.com, ght@cos.ufrj.br

Abstract. *This work presents a model inspired in the Laws of Software Evolution and supported by techniques of System Dynamics, whose objective is to enable the construction of computational infrastructure to simulate how software systems behave throughout successive cycles of maintenance in order to allow better understanding of the decaying process.*

Resumo. *Este trabalho apresenta um modelo inspirado nas Leis de Evolução de Software e apoiado por técnicas de Dinâmica de Sistemas, cujo objetivo é possibilitar a construção de infraestrutura computacional que permita simular como sistemas de software se comportam ao longo de sucessivos ciclos de manutenção, de forma a proporcionar uma melhor compreensão do processo de decaimento.*

1. Introdução

Sistemas de software normalmente passam por sucessivos ciclos de manutenção. Em muitas situações, isso acaba contribuindo para o decaimento de software, caracterizado pela deterioração da estrutura de um sistema em evolução, afetando sua qualidade e dificultando a realização de novas manutenções (EICK *et al*. 1999). Isso se agrava considerando-se que atualmente os sistemas oferecem quantidades crescentes de funcionalidades e, consequentemente, tendem a ter maior complexidade e tamanho, o que dificulta ainda mais o processo de manutenção. Uma maior compreensão dos caminhos pelos quais sistemas evoluem permite que sejam modificados com maior facilidade e sua qualidade não seja prejudicada.

Nessa direção, experiências vivenciadas na gerência e no desenvolvimento de sistemas sujeitos a grande quantidade de modificações e utilizados por diversos usuários, motivaram este trabalho. Esses sistemas estavam associados a altos custos com manutenção, fazendo até mesmo que modificações inicialmente menos complexas demandassem grande volume de tempo e recursos. Apesar do grande investimento realizado na construção desses sistemas, em função dos altos custos de manutenção, indicava-se a necessidade de sua reestruturação ou até mesmo sua descontinuidade e substituição por outros. Por outro lado, pressões relativas a custos e prazos faziam com que normalmente as manutenções em software fossem feitas de forma desorganizada, isolada e com baixa qualidade, o que comprometia mais ainda a qualidade e vida útil desses sistemas, que cresciam rapidamente em termos de funcionalidade, tamanho e estrutura. Dessa forma, tornou-se uma motivação procurar entender os mecanismos pelos quais um software é submetido quando em processo de manutenção, de maneira a acomodar melhor as modificações que necessitam ser realizadas. Acreditamos que observar esse processo de evolução poderia auxiliar na acomodação de mudanças, proporcionando maior qualidade e

manutenibilidade nas futuras versões de um produto de software, estendendo sua vida útil e, conseqüentemente, aproveitando de maneira mais eficiente o investimento realizado.

A literatura técnica tem apresentado diversos trabalhos nessa área, onde praticamente todos eles abordam evolução de software em nível conceitual ou através da observação de evolução de versões de um sistema em particular, considerando o código fonte de sistemas legados ou software livre (ARAÚJO 2009). SMITH e RAMIL (2002), por outro lado, apresentam um modelo inicial de Dinâmica de Sistemas para Evolução de Software, através de análise qualitativa, porém sem ferramental de apoio para a execução deste modelo. STOPFORD e COUNSELL (2008) apresentaram, recentemente, o único trabalho, que pudemos encontrar na literatura técnica, descrevendo um *framework* para simulação de evolução de software, baseado em um processo de simulação fictício e na geração automática de um conjunto de requisitos e código fonte, processados através de agentes que implementam políticas de evolução definidas pelo usuário.

Dentro dessa realidade, 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 a observação da evolução do software representam importantes pontos de partida para uma melhor compreensão do processo de decaimento de sistemas. Reconhecendo a necessidade de investigações neste contexto, este trabalho apresenta a versão evoluída de um modelo (ARAÚJO e TRAVASSOS 2006) e uma infraestrutura para observar a evolução de software através de modelos dinâmicos calibrados com dados históricos do projeto de software, 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 decaimento.

Este trabalho é organizado em mais três seções, além dessa introdução. Na seção 2 é apresentada a construção de um modelo para observação de evolução de software. A seção 3, baseada nesse modelo, apresenta uma infraestrutura para observação de evolução de software. A seção 4, por sua vez, utiliza-se dessa infraestrutura para avaliar o modelo construído, a partir de uma prova de conceito. Finalmente, a seção 5 apresenta as conclusões e perspectivas futuras deste trabalho.

2. Modelo para Observação de Evolução de Software

No sentido de observar o comportamento de sistemas de software em evolução, de forma a possibilitar um maior conhecimento desse processo, objetivo principal deste trabalho, esta seção apresenta a construção de um modelo cujo ponto de partida baseia-se nas Leis de Evolução de Software (LSE – *Laws of Software Evolution*) (LEHMAN 2003), que descrevem como um sistema se comporta através de sucessivas versões, considerando o seu envelhecimento, representado pelo decaimento da qualidade estrutural.

Para isso, foram selecionadas características de software que pudessem representar aspectos de qualidade desses sistemas e que capturassem a essência do comportamento evolutivo associado a cada LSE. A norma ISO 9126-1[1] foi o ponto de partida para esta seleção e, dessa norma, foram adaptadas características como 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). Entretanto, essas características não são suficientes para cobrir os aspectos considerados pelas Leis de Evolução de

[1] Tal norma encontra-se em revisão e, a princípio, não há indícios que essa revisão irá modificar os elementos principais que foram utilizados no contexto deste trabalho.

Software. A partir da conceituação teórica proposta para cada LSE, novas características foram identificadas, representando 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) e Esforço (considerada como o montante de trabalho realizado, medido em termos de homens/hora ou unidade equivalente), importantes para observar questões relativas à evolução de software (ARAÚJO *et al*. 2005).

No intuito de observar a evolução do produto de software como um todo, torna-se necessário avaliar o impacto entre as características de software consideradas. As premissas apresentadas para a observação do comportamento de sistemas em evolução partem do princípio de que as características de software influenciam umas às outras. A partir do conhecimento tácito sobre o comportamento dessas características, foi construído um modelo inicial para a observação de evolução de software, apresentado em ARAÚJO e TRAVASSOS (2006), utilizando-se um Diagrama de Causas e Efeitos (FORRESTER 1961), descrevendo os comportamentos as características através de suas tendências (aumento, diminuição, constância). Entretanto, o modelo apresentaria ameaça à sua validade ao afirmar a existência de influências entre as características de software consideradas sem um estudo mais aprofundado. Assim, foi realizado um conjunto de estudos baseados em revisão sistemática (KITCHENHAM 2004) (BIOLCHINI *et al.* 2005) com o objetivo de investigar as influências entre todas as características de software. Mais recentemente, este tipo de revisão sistemática foi identificada como sendo uma *quasi*-revisão sistemática (TRAVASSOS *et al*. 2008). Dessa forma, foi construído um metamodelo para essas revisões, buscando observar a relação entre cada par das características de software (ARAÚJO e TRAVASSOS 2008), através de questões que visavam identificar a existência de influência entre duas características de software, qual a direção dessa influência e qual a intensidade/taxa da influência identificada. Para responder a tais perguntas, buscas em bibliotecas digitais foram conduzidas. Eventualmente, trabalhos podem ser mais ou menos precisos nas suas abordagens, coleta de dados, abordagens estatísticas ou, até mesmo, podem se contradizer. Através de uma classificação dos artigos por meio de um *score* calculado a partir desses elementos, quando conflitantes, pôde-se selecionar os trabalhos a serem utilizados. Baseado neste cenário, a Figura 1 apresenta os relacionamentos de influência entre as características de software que foram considerados na construção do modelo para observação de evolução de software, tendo todas essas relações apoiadas pela literatura técnica (ARAÚJO 2009). As setas indicam o efeito que uma característica provoca sobre a outra, assinalando se este efeito aumenta (sinal positivo), reduz (sinal negativo) ou alterna (sinais positivo/negativo) a presença da característica de destino.

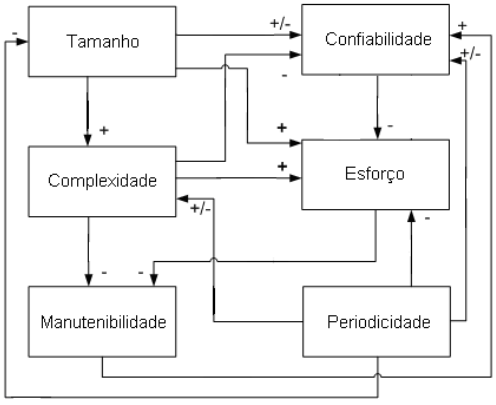

Figura 1. Diagrama de Causas e Efeitos do Modelo para Observação de Evolução de Software

3. Infraestrutura para Observação de Evolução de Software

O comportamento de sistemas de software não é estático ao longo de seu ciclo de vida. Assim, através da identificação das ligações e influências entre as LSE, é possível a construção de modelos dinâmicos, no sentido de simular para observar o comportamento de sistemas de software, através de sucessivos ciclos de manutenção.

Desta forma, uma infraestrutura foi construída compreendendo um conjunto de atividades para sua operacionalização. Inicialmente, o Engenheiro de Software coleta dados de um sistema real, no qual deseja observar o processo de evolução. Essas informações podem ser utilizadas para gerar as equações utilizadas no processo de simulação, através de regressão linear que, neste trabalho, são geradas a partir de uma planilha eletrônica. Justifica-se o uso de regressão linear, apesar da possibilidade de uma maior margem de erro, em função de observações dessa natureza normalmente serem realizadas através de análise semiquantitativa para a análise de dados, através da tendência das observações, mais do que a precisão dos valores reais. O conjunto de equações geradas pode ser computado através de uma ferramenta que ofereça apoio a modelos dinâmicos como, por exemplo, Dinâmica de Sistemas. Por fim, pode-se observar o resultado da simulação e o processo de decaimento de software, indicando ainda o estado de cada Lei de Evolução de Software em função das tendências de comportamento das características de software e do modelo considerado. Esse processo está apresentado na Figura 2 e oferece subsídios ao Engenheiro de Software para interferir no processo de desenvolvimento e manutenção de software, visando minimizar seu decaimento (ARAÚJO 2009).

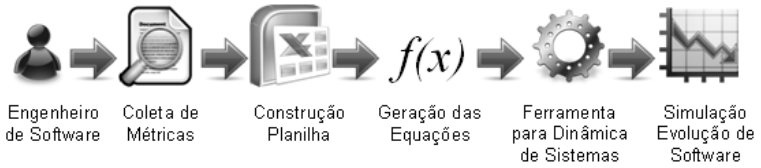

Figura 2. Processo para Observação de Evolução de Software

Para apoiar a execução de modelos baseados em Dinâmicas de Sistemas, foi construída uma extensão da ferramenta Illium (BARROS 2001), chamada Illium Software Evolution (ARAÚJO 2009), evidenciada na Figura 3, que apresenta a arquitetura para a infraestrutura proposta.

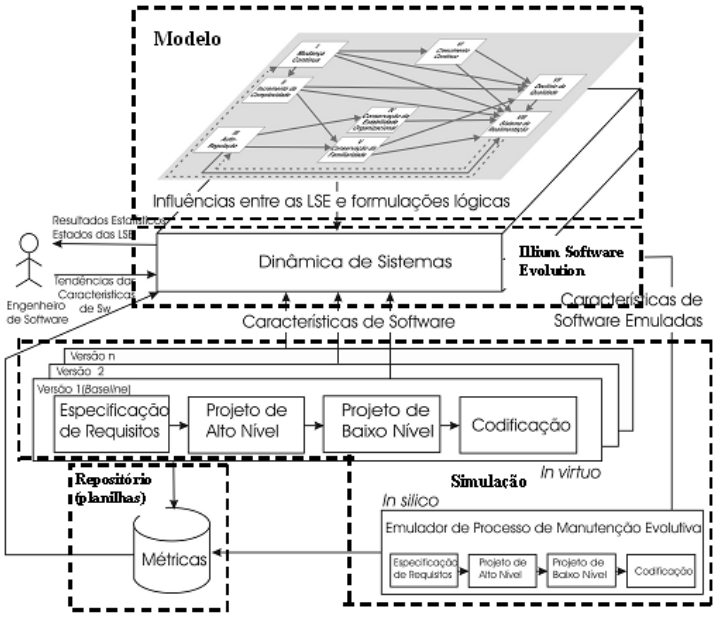

Figura 3. Arquitetura da infraestrutura para observação de evolução de software

4. Avaliação da Infraestrutura

Para avaliação dessa infraestrutura, uma prova de conceito é apresentada (ARAÚJO 2009), voltada para uma aplicação do domínio de planejamento e gestão. Trata-se de uma aplicação Web 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. 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. Foram consideradas 16 versões do sistema, disponíveis nos repositórios de controle de versões, controle de defeitos e esforço utilizado, cujas medidas são relevantes para a observação da evolução do sistema. A Tabela 1 apresenta o conjunto das medidas coletadas, enquanto a Figura 4 apresenta o comportamento individual das características de software.

Tabela 1. Medidas coletadas do sistema em observação

#	Tam	Period	Compl	Esf	Conf	Manut		#	Tam	Period	Compl	Esf	Conf	Manut
1	62,183	0	11,3384	6	1	0		9	62,572	1	11,4016	12	3	9
2	62,578	1	11,4076	0,25	1	0,25		10	62,605	7	11,4016	6	3	3
3	62,524	6	11,3913	30,75	7	30,75		11	62,892	4	11,4226	1,75	3	1,75
4	62,537	2	11,3913	3,75	2	3,75		12	63,296	0	11,4297	0,5	1	0,5
5	62,551	7	11,3929	1	1	0		13	63,682	20	11,5099	1,7	3	1,7
6	62,551	1	11,3929	14	2	14		14	63,703	2	11,5131	1	1	1
7	62,578	5	11,4016	2	2	1		15	63,692	6	11,5131	3	2	3
8	62,566	1	11,4016	4	3	4		16	63,726	17	11,5196	14	6	10

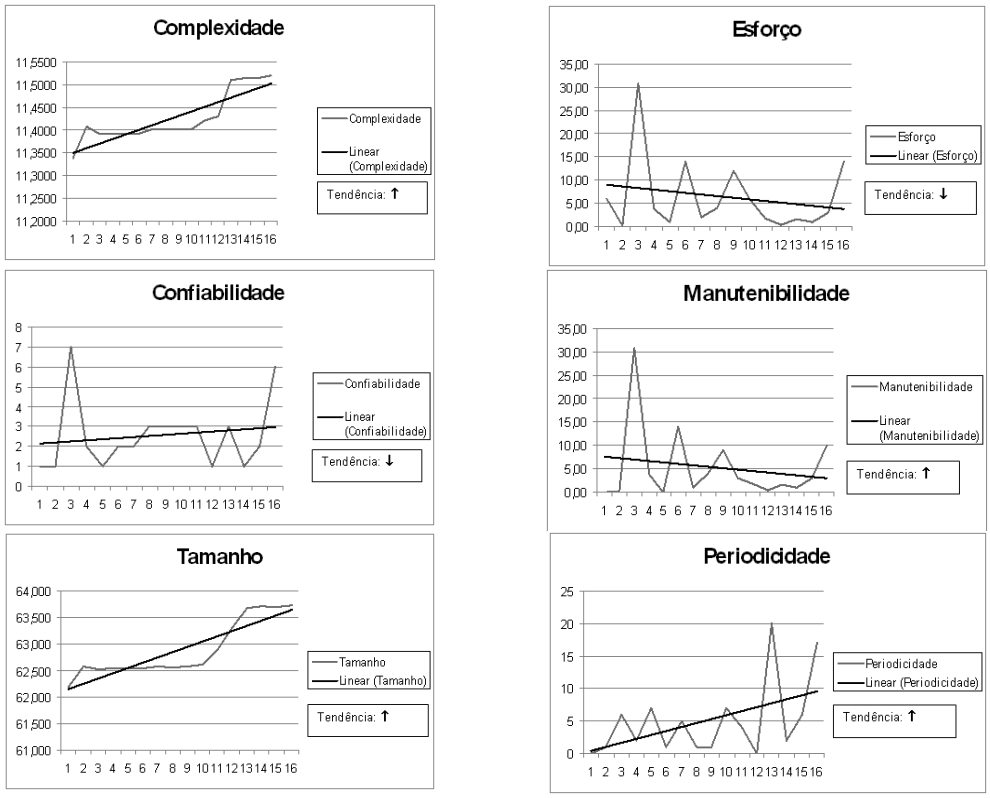

Figura 4. Comportamento das características de software

Nessa figura observa-se um comportamento de tendência de aumento das características Tamanho, Periodicidade e Complexidade. Para periodicidade, isso não seria necessariamente um problema, uma vez que podem acontecer iterações mais longas no desenvolvimento ou manutenção de um produto de software. Observa-se ainda uma variação significativa da característica Esforço ao longo das versões analisadas, com tendência de declínio. A figura também apresenta o comportamento da característica

Confiabilidade, medida através do número de defeitos identificados por versão, indicando uma tendência de diminuição, em função do aumento do número de defeitos. Por fim, o comportamento da característica Manutenibilidade apresenta tendência de aumento em função da diminuição do esforço gasto na correção de defeitos.

Considerando esse conjunto de medidas e as influências entre as características de software, foram construídas as equações das curvas, através de regressão linear, e com o apoio de uma planilha construída especificamente para esse fim. A listagem a seguir apresenta um fragmento das equações geradas para esse conjunto de dados. No contexto de Dinâmica de Sistemas, um PROC significa uma equação do modelo, formada pela variável a ser calculada, e seguida pela expressão propriamente dita.

```
PROC PERIODICITY 0.227272727272727 * TIMER;
PROC SIZE 0.0256675126903551 * PERIODICITY;
PROC COMPLEXITY 0.0706132821742326 * SIZE + 0.00251206218274111 * PERIODICITY;
PROC MAINTAINABILITY -11.9523440794294 * COMPLEXITY;
PROC RELIABILITY -0.923130260826362 * SIZE + 0.199238578680203 * PERIODICITY + 2.23944363740078 *
    COMPLEXITY + 0.167732091652856 * MAINTAINABILITY;
PROC EFFORT -8.17701759613598 * SIZE + 0.389593908629441 * PERIODICITY + -75.3565134753427 *
    COMPLEXITY + 4.54464285714286 * RELIABILITY;
```

A partir da execução da ferramenta Illium Software Evolution (Figura 5), pode-se observar, o comportamento exibido pelo sistema em evolução. A manutenibilidade não apareceu no gráfico em função da escala e, através dos dados gerados, é possível verificar que apresenta tendência de se manter constante.

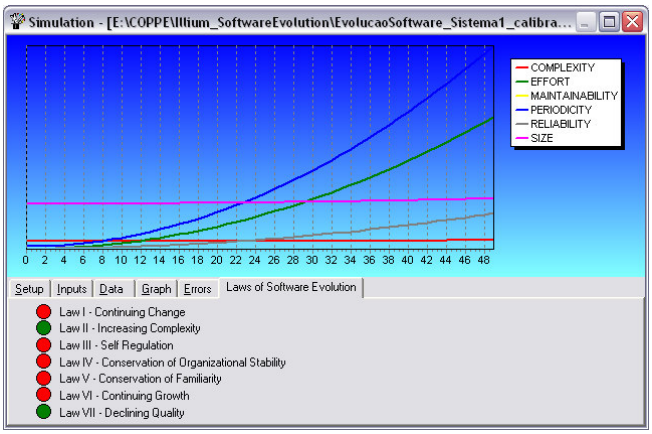

Figura 5. Simulação do comportamento das LSE

O gráfico apresenta o comportamento de simulação de cada uma das características de software, em função do modelo considerado e das equações construídas. É importante ressaltar que o eixo X apresenta o número de versões e, o eixo Y, o comportamento de cada característica de software. Para o eixo Y, como as características são apresentadas em unidades distintas, não é apresentada a escala. Disso se intui que a distância entre as curvas é pouco representativa. O objetivo é mostrar o comportamento de cada uma das características de software, numa análise semiquantitativa. A ferramenta ainda apresenta o comportamento do sistema para cada Lei de Evolução de Software, onde um semáforo é associado a cada LSE, sendo que a cor verde significa que a respectiva Lei alcançou sua implicação positiva (as Leis II e VII), e a vermelha (as demais Leis), a implicação negativa. Uma implicação positiva revela que a Lei pode resultar em uma situação oposta ao decaimento de software, como a diminuição da complexidade estrutural através do uso de técnicas antirregressivas, por exemplo. Uma implicação negativa é aquela que resulta em decaimento de software e descreve os interesses desta pesquisa, como a perda de qualidade em um sistema ou o aumento de sua complexidade. A Tabela 2 apresenta um possível comportamento esperado para a aplicação, baseado nas tendências das medidas coletadas e considerando que o comportamento do sistema se mantenha nas versões seguintes,

comparado com o comportamento simulado de versões futuras. Pode-se observar uma nova tendência de comportamento para o sistema, com mudança das características esforço e manutenibilidade, sombreado em cinza.

Tabela 2. Comportamento das características de software após a simulação

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

A Tabela 3 apresenta a análise do comportamento das Leis de Evolução de Software após o processo de simulação, onde se pode observar o estado de cada LSE.

Tabela 3. Análise do comportamento das LSE

LSE	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	Apesar da manutenibilidade se mostrar constante, o esforço e a complexidade estão aumentando, o que pode indicar uma perda de qualidade do sistema no futuro.

No sentido de avaliar esse comportamento futuro do sistema em evolução, foram coletadas as medidas de mais oito versões, posteriores às coletadas para a construção das equações para a simulação. A Tabela 4 apresenta as tendências de comportamento para cada característica de software, considerando o comportamento esperado, em função das versões utilizadas para a construção do modelo, o comportamento simulado e aquele observado nas novas versões do sistema.

Tabela 4. Comparativo entre os comportamentos esperado, simulado e observado

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

Observa-se que as tendências previstas de aumento para Tamanho, Periodicidade e Complexidade foram mantidas para essas novas versões. O esforço apresenta uma situação peculiar. As medidas coletadas nas primeiras versões do sistema indicavam uma tendência de diminuição. Entretanto, após o processo de simulação, pôde-se observar uma inversão nesse comportamento, com tendência de aumento, o que pôde ser constatado nas novas versões coletadas. A confiabilidade manteve sua tendência de diminuição, em função do aumento do número de defeitos. Por fim, a manutenibilidade, que apresentava inicialmente uma tendência de aumento, passou a tender para estabilização em versões futuras. Apesar da tendência de aumento em função do menor esforço em manutenção, os dados coletados nessas novas versões apresentam uma inclinação menor do que foi inicialmente observado, o que pode indicar uma situação de estabilidade futura, conforme previsto na simulação. A partir dessas observações, ações podem ser tomadas para diminuir o processo de

decaimento do sistema em questão, principalmente em relação às LSE que atingiram a implicação negativa. Em relação à periodicidade estar aumentando, o que pode indicar que o sistema que não está mais em mudança, a entrega de novas versões em menor intervalo de tempo pode minimizar essa questão, através da diminuição de cada iteração. A característica confiabilidade apresenta uma tendência de diminuição, ou seja, aumento do número de defeitos. Uma ação a ser tomada é enfatizar o uso de técnicas de Verificação, Validação e Testes, no sentido de garantir uma maior qualidade da aplicação em evolução. O aumento do esforço, que pode indicar uma falta de estabilidade organizacional, pode ser abordado através de iniciativas que minimizem a rotatividade na equipe, ou promover uma maior capacitação dos profissionais. O aumento da complexidade pode ser controlado com um gerenciamento mais fino da complexidade estrutural do sistema, apontando para a utilização de técnicas de gerenciamento da complexidade e melhoria da confiabilidade.

5. Conclusões e Perspectivas Futuras

Este trabalho apresentou um modelo conceitual e uma infraestrutura para observação de evolução de software, baseado em Dinâmica de Sistemas, utilizando dados extraídos de um sistema real. Essa infraestrutura pode ser uma ferramenta importante no processo de tomada de decisão por parte do Engenheiro de Software, através da simulação do comportamento futuro do sistema em observação. Isso pode ajudar na aplicação de manutenções preventivas e, consequentemente, melhorar a qualidade e manutenibilidade de sistemas, ou no planejamento de sua descontinuidade. Os resultados obtidos abrem novas perspectivas de pesquisa e, apesar de ser uma atividade que consome uma grande quantidade de recursos no ciclo de vida, manutenção ainda é uma atividade que carece de um melhor entendimento de como é realizada. Nesse sentido, novas pesquisas estão sendo realizadas, sendo objeto de futuras investigações e publicações, no sentido de construir uma base de conhecimento sobre esse assunto, através da observação de como desenvolvedores executam atividades de manutenção, objetivando tratar prioritariamente do problema de entendimento do processo de manutenção de software.

Referências

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., TRAVASSOS, G.H., 2006, "Estudos Experimentais em Manutenção de Software: Observando Decaimento de Software Através de Modelos Dinâmicos". *WMSWM'06*.

ARAÚJO, M.A.P., TRAVASSOS, G.H., 2008, *A System Dynamics Model based on Cause and Effect Diagram to Observe Object-Oriented Software Decay*, Technical Report ES-720/08 -COPPE/UFRJ.

ARAÚJO, M.A.P., 2009, *Um Modelo para Observação de Evolução de Software*. Tese de Doutorado, COPPE/UFRJ.

BARROS, M.O., 2001, *Gerenciamento de Projetos Baseados em Cenários: Uma Abordagem de Modelagem Dinâmica e Simulação*. Tese de Doutorado, COPPE/UFRJ.

BIOLCHINI, J.; MIAN, P. ; CONTE, T. U. ; NATALI, A. C. C. ; TRAVASSOS, G. H., 2005. *Lessons Learned on Applying Systematic Reviews to Software Engineering*. In: WSESE2005.

CAMILETTI, G.G., FERRACIOLI, L., 2002, "The Use of Semiquantitative Computational Modelling in the Study of Predator-Pray System". In: *X International Organization for Science and Technology Education*.

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.

FORRESTER, J.W., 1961, Industrial Dynamics, Cambridge, MA: The MIT Press

KITCHENHAM, B., 2004, Procedures for Performing Systematic Reviews, Joint Technical Report, Keele University.

LEHMAN, M.M., RAMIL, J.F., 2003, "Software Evolution – Background, Theory, Practice". *Information Processing Letters*, vol. 88, pp. 33 – 44.

SMITH, N., RAMIL, J.F., 2002, "Qualitative Simulation of Software of Evolution Process". In: *WESS´02 Eighth Workshop on Empirical Studies of Software Maintenance*.

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.

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.