Otimizando recursos no processo de inspeção de software: uma abordagem utilizando simulação com dinâmica de sistemas

Jailton S. Coelho[1], José Luis Braga[1], Bernardo Giori Ambrósio[2]

[1] Departamento de Informática,
Universidade Federal de Viçosa, Brasil
jailton.coelho@ufv.br, zeluis@dpi.ufv.br
[2] Departamento de Ciências Exatas e Aplicadas,
Universidade Federal de Ouro Preto, Brasil
bernardo@decea.ufop.br

Abstract. Reparar um defeito de software pode custar até 100 vezes mais caro, caso ele não seja encontrado o mais próximo possível de onde foi cometido. A inspeção de software é uma técnica que pode ser usada para ajudar a detectar defeitos nas fases iniciais do processo de desenvolvimento, evitando que esses defeitos sejam propagados para as fases seguintes. O custo/benefício de inspeções pode se tornar bastante significativo, caso as inspeções sejam realizadas de forma eficiente. Por ser influenciada por muitos fatores de qualidade, a análise do contexto da inspeção como um todo pode se tornar complexa. Gerentes de projeto deixam de utilizar a inspeção por falta de dados e estudos sobre os reais benefícios que ela pode gerar. Este artigo apresenta um modelo de dinâmica de sistemas, para apoiar decisões sobre esse ponto de vista. As variáveis que fazem parte do modelo são quantificadas com base em experimentos reais ou empíricos disponibilizados na literatura, tornando os resultados do modelo próximos do que seria obtido no mundo real. O modelo permite reproduzir cenários nos quais seria caro ou perigoso experimentar no mundo real, sendo possível antever os impactos que a inspeção pode trazer no processo de desenvolvimento.

Keywords: inspeção de software, dinâmica de sistemas, detecção de defeitos

1 Introdução

A qualidade de um produto de software não pode ser introduzida após a finalização do mesmo, deve ser acompanhado e garantido durante todo o processo de desenvolvimento [16]. A participação humana é um fator presente nos projetos de software, o que justifica a ocorrência de defeitos nos sistemas de software mesmo quando as melhores técnicas e procedimentos são empregados no seu desenvolvimento [12].

Em torno de 40 a 50% do esforço no desenvolvimento de software é gasto em retrabalho e grande parte desse esforço é gasto com a remoção de defeitos, exigindo,

em algumas situações, mudanças significativas na arquitetura do projeto [17]. O custo para remover um defeito de software aumenta na medida em que o processo de desenvolvimento progride para os estágios finais [17]. Encontrar e corrigir um defeito de software pode ser até 100 vezes mais caro, caso ele não seja encontrado próximo da fase em que foi cometido no processo de desenvolvimento [17]. [13] considera defeito como qualquer desvio nas propriedades de qualidade do software, o que inclui o não atendimento aos requisitos estabelecidos.

Atividades de prevenção e detecção de defeitos devem ser introduzidas ao longo de todo o processo de desenvolvimento com o objetivo de diminuir os custos de correção nas fases finais. Alocar recursos para a realização de revisões de artefatos de software, tão logo estejam finalizados, pode contribuir para reduzir o retrabalho e melhorar a qualidade dos produtos [6]. Encontrar um defeito o mais cedo possível pode gerar uma enorme contribuição para o projeto, do ponto de vista econômico e de qualidade.

A inspeção de software é um tipo específico de revisão que pode ser aplicado em artefatos de software e possui um processo de detecção de defeitos rigoroso e bem definido [15]. Segundo [13], inspeções têm a função de detectar defeitos antes que a fase de teste seja iniciada, contribuindo para a melhoria da qualidade geral do software em relação ao seu orçamento e benefícios de tempo. Apesar desses benefícios, em geral os gerentes de projetos não adotam a inspeção no processo de desenvolvimento de software devido ao receio de não obterem um retorno satisfatório [6], que decorre da falta de ferramentas para auxiliar na obtenção de uma estimativa do retorno que será obtido com a alocação de recursos para realizar tais atividades.

Esse artigo apresenta um modelo de dinâmica de sistemas [7, 14, 18] que permite avaliar os impactos da alocação de recursos para a realização de inspeções em várias fases do processo de desenvolvimento de software. O modelo permite simular e analisar os resultados da política gerencial quanto à realização de inspeções antes de aplicá-la em um projeto real. O modelo pode ser usado como uma ferramenta de apoio à decisão permitindo simular diversos cenários de acordo com as características da empresa, equipe e projeto, sem que seja necessário realizar experimentos em projetos reais, o que seria caro e demandaria tempo.

No modelo foram incluídas variáveis que influenciam diretamente na qualidade do processo de inspeção, bem como no retorno obtido ao alocar recursos para a sua realização, dentre as quais tem-se: tamanho da equipe de inspetores, técnica de leitura, nível CMMI da empresa, tipo e tamanho do documento, entre outras. Essas variáveis e os relacionamentos de influência existentes entre as mesmas foram identificados e calibrados com base em experimentos reais ou empíricos publicados na literatura, o que garante a consistência do modelo e o torna coerente com os conhecimentos difundidos pela Engenharia de Software.

Este artigo está estruturado como se segue: a Seção 2 aborda os conceitos envolvidos no contexto desse trabalho e descreve alguns trabalhos correlatos, a Seção 3 descreve o processo de construção do modelo, a Seção 4 apresenta as simulações realizadas com o modelo construído para avaliar os níveis de influência dos fatores que causam impacto na atividade de inspeção e a Seção 5 descreve as conclusões e os trabalhos futuros.

2 Contexto

Este trabalho classifica-se como pesquisa aplicada ou tecnológica, pois utiliza conhecimentos básicos, disponíveis na literatura, para gerar novos conhecimentos para aplicação prática. Esse tipo de pesquisa se baseia em um estudo observacional e, no âmbito dos procedimentos de pesquisa, o mesmo é caracterizado pela extensa pesquisa bibliográfica [9].

2.1 Inspeção de Software

A inspeção é um tipo particular de revisão que contribui para garantir a qualidade do produto de software. Todas as etapas do processo de desenvolvimento de software são suscetíveis à incorporação de defeitos, que podem ser detectados pela inspeção e posteriormente removidos.

O processo básico da inspeção consiste de seis etapas consecutivas [11]:

- **Planejamento**. Desempenhado por um moderador cuja função é definir o contexto da inspeção (descrição da inspeção, técnica de leitura a ser utilizada na atividade de detecção de defeitos, documento a ser inspecionado, autor do documento, dentre outros), selecionar os inspetores e distribuir o material a ser inspecionado.
- **Detecção de Defeitos**. Os inspetores selecionados têm a tarefa de encontrar defeitos no documento entregue a eles e, ao longo da inspeção, produzir uma lista dos possíveis defeitos.
- **Coleção de Defeitos**. O moderador tem a função de agrupar as listas de defeitos dos inspetores, eliminando os defeitos repetidos, ou seja, que foram encontrados por mais de um inspetor, mantendo só um registro para cada defeito.
- **Discriminação de Defeitos**. O moderador juntamente com o autor do documento e os inspetores discutem os defeitos a fim de classificá-los como defeito de fato ou falso positivo. Os falsos positivos são eliminados da lista de defeitos, permanecendo apenas os defeitos reais.
- **Retrabalho**. Os defeitos que fazem parte da lista são corrigidos pelo autor do documento e ao término é produzido um relatório de correção dos defeitos.
- **Continuação**. É decidido pelo moderador se uma nova inspeção deve ou não ser realizada.

A inspeção utiliza-se da revisão baseada na leitura e entendimento de um artefato de software a fim de encontrar defeitos. Muitos desenvolvedores possuem habilidades para escrever documentos de requisitos e códigos de software, mas não é oferecido a eles nenhum suporte relacionado à leitura e revisão adequada desses artefatos. Uma das soluções é então fazer uso de técnicas de leitura.

Técnicas de leitura contribuem para o aumento da compreensão sobre algum artefato de software. Esta compreensão deve ser suficiente a ponto de permitir que os inspetores identifiquem tanto as informações importantes para a execução de uma determinada tarefa, como a relação destas informações com o problema que está sendo tratado [6]. Algumas das principais técnicas de leitura são [13]:

- **Ad-hoc**: não oferece nenhum suporte técnico de como detectar defeitos em um artefato de software. Neste caso, a detecção de defeitos depende totalmente da habilidade, do conhecimento e da experiência de um inspetor.
- **Checklist (LBCh)**: baseia-se em uma série de perguntas (frequentemente questões cuja resposta é sim ou não) sobre assuntos do documento a ser inspecionado.
- **Cenário (LBCe)**: faz uso da noção de cenários que fornecem orientação personalizada para inspetores sobre como detectar defeitos. Um cenário pode ser um conjunto de perguntas ou uma descrição mais detalhada para um inspetor sobre a forma de realizar a análise do documento.
- **Stepwise Abstraction (SA)**: fornece instruções de leitura mais estruturadas e precisas para os artefatos de código. Ao ler uma sequência de declarações no código, o inspetor abstrai uma função de calcular dessas declarações. Este procedimento é repetido até que a última função do artefato de código inspecionado seja captada, comparando no final essas funções com a especificação.
- **Análise de Pontos de Função (APF)**: define um sistema em termos de suas entradas, arquivos, consultas e saída. Os cenários de pontos de função são desenvolvidos em torno desses itens. Um cenário de pontos de função consiste em perguntas e direciona o foco de um inspetor para um item de ponto de função específico dentro do documento de requisitos inspecionado.
- **Perspectiva (LBPe)**: esta técnica baseia-se na ideia de que um produto de software deve ser inspecionado a partir da perspectiva dos diferentes stakeholders. As perspectivas desta técnica são elaboradas de modo que os inspetores são forçados a enxergar o ponto de vista do usuário, do projetista e do testador do sistema, utilizando a perspectiva correspondente.
- **N-fold**: o método de leitura N-fold consiste na replicação do processo de realização de inspeções formais para detecção de defeitos usando N equipes trabalhando em paralelo com um único moderador que é o responsável por coordenar e reunir os resultados. Essa técnica é aplicada considerando a hipótese de que a existência de equipes diferentes melhora a eficiência do processo, detectando defeitos que não seriam encontrados por uma única equipe de inspetores.

2.2 Dinâmica de Sistemas

A dinâmica de sistemas é uma técnica descritiva utilizada na modelagem e simulação de sistemas, que permite levar em consideração as variações dinâmicas dos problemas, sendo fortemente baseada no pensamento e análise sistêmicos e na teoria matemática dos Sistemas Dinâmicos. A Dinâmica de Sistemas permite a compreensão de como as políticas adotadas, ou a própria estrutura do sistema, afetam ou determinam o seu comportamento dinâmico, permitindo prever comportamentos futuros e possíveis colapsos. O uso dessa técnica permite a análise e compreensão de problemas e situações de maneira integrada e interconectada, deixando claras as relações existentes entre as variáveis de decisão [2].

Um modelo de dinâmica de sistemas possui dois elementos principais: os estoques, que são os recursos acumuláveis do sistema, e os fluxos ou taxas, que são funções que representam as decisões ou políticas da empresa com relação aos usos e acúmulos dos estoques ou recursos. Há também os conversores, ou variáveis simples, que são os

elementos do modelo que exercem influência sobre os valores dos fluxos [14]. Esses elementos são ilustrados na Fig. 1.

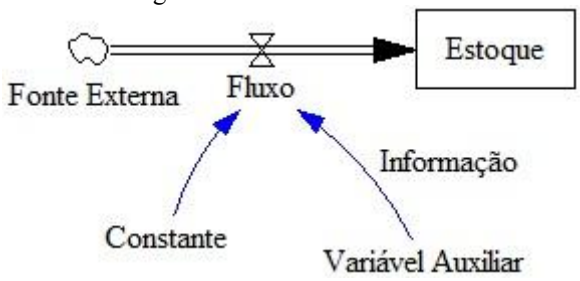

Fig. 1. Elementos básicos de um modelo genérico de estoque e fluxo [14].

Segue a descrição dos elementos da Fig. 1:

- Estoques: representam os acúmulos ou decrementos ao longo do tempo. Seus valores são influenciados pelos fluxos de entrada e/ou de saída.
- Fonte externa: são recursos ou repositórios infinitos utilizados pelos fluxos que não são especificados no modelo e não fazem parte do contexto da simulação.
- Fluxos: podem ser considerados as "ações" no sistema. Eles influenciam diretamente os estoques e podem representar as decisões e políticas adotadas.
- Conversores (ou variáveis auxiliares): são utilizados na elaboração dos detalhes da estrutura do modelo e para representar as variáveis que influenciam os fluxos. Eles se relacionam com os fluxos através de setas unidirecionais.
- Informação (setas unidirecionais): são utilizadas para representar fluxos de informação e interligar as variáveis do modelo. Estas setas permitem construir loops de realimentação (feedback) no modelo [18].

2.3 Trabalhos correlatos

Um dos modelos de dinâmica de sistemas mais difundidos na área de Engenharia de Software é o modelo proposto por [1] para simular o desenvolvimento de projetos de software. Esse modelo considera os seguintes aspectos de um projeto: Gerência de Recursos Humanos, Produção de Software, Desenvolvimento de Software, Garantia de Qualidade e Retrabalho, Testes e Controle.

[14] usou a disciplina dinâmica de sistemas com o objetivo de modelar os efeitos da realização de inspeções formais em diversas fases do ciclo de desenvolvimento de softwares. Foram comparados questões como impactos de custo, prazo e qualidade para diferentes taxas de inspeção. O modelo trata também da correlação entre fluxos e tarefas, erros e recursos humanos entre as diversas fases do desenvolvimento de projetos de software.

O modelo proposto neste artigo também utiliza a dinâmica de sistemas. Entretanto, o modelo proposto simplifica a visão dos fatores que mais influenciam na qualidade e eficiência de inspeções, permite aos usuários interagir com o modelo e é focado na etapa de detecção de defeitos, que é uma das seis etapas da atividade de inspeção.

3 Construção do Modelo

O modelo de dinâmica de sistemas apresentado neste trabalho foi construído seguindo o método proposto por [2], constituído por três etapas subdivididas em um ou mais passos. Na primeira etapa são identificadas as variáveis envolvidas no processo ou problema que se pretende modelar. Em seguida, os relacionamentos existentes entre essas variáveis devem ser identificados, o que pode ser feito com o auxílio dos diagramas de influência, que permitem construir modelos simples e de fácil entendimento e são muito utilizados em análise sistêmica. Os diagramas de influência permitem modelar relações de influência entre as variáveis, facilitando a identificação dos laços de realimentação [18]. Na segunda etapa os diagramas de influência são mapeados em modelos de dinâmica de sistemas seguindo algumas regras estabelecidas no método proposto. Na terceira etapa, os modelos de dinâmica de sistemas são refinados e os relacionamentos entre seus elementos são quantificados e testados, até que seja obtido um modelo finalizado e validado.

3.1 Seleção das Variáveis

Inicialmente foram identificadas as variáveis que representam os fatores que mais influenciam a atividade de inspeção de artefatos de projetos de software, que segundo [13] são: as características do time, o esforço, a técnica de leitura, a organização da atividade de detecção de defeitos e as características do produto. Essas variáveis são ilustradas na Fig. 2.

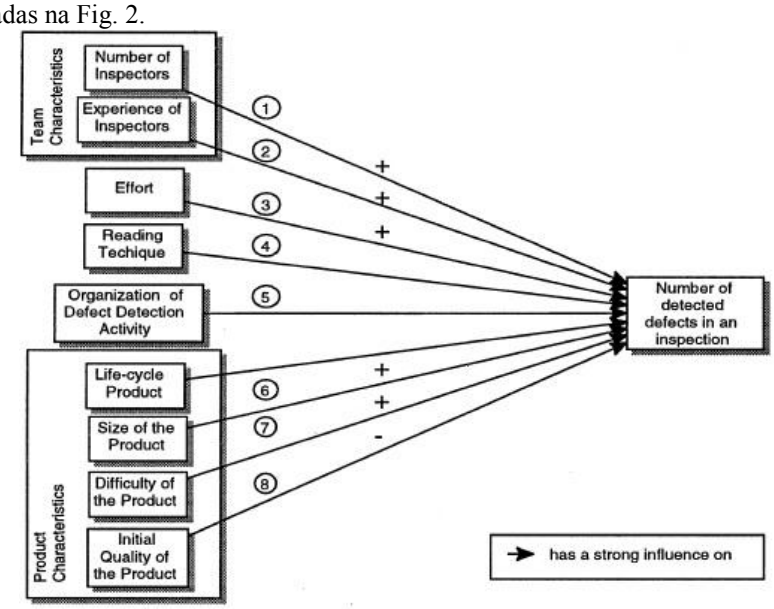

Fig. 2. Principais fatores de forte influência no desempenho do processo de inspeção [13].

As características do time referem-se à quantidade e à experiência dos inspetores envolvidos no processo. O esforço representa o custo total necessário para realização das inspeções. A técnica de leitura, que pode ser diferente em cada fase do ciclo-de-vida do produto, refere-se ao modo como será realizada a leitura do artefato para detectar os defeitos. A organização da atividade de detecção de defeitos refere-se ao número de inspeções a serem realizadas, à técnica de leitura e ao número de inspetores que farão parte da equipe. As características do produto são representadas pelo ciclo de vida, tamanho, complexidade e qualidade inicial do produto. O ciclo de vida se refere à fase de desenvolvimento do projeto. O tamanho do documento se refere ao número de páginas que ele possui. A dificuldade do produto se refere à complexidade de entendimento do mesmo, ou seja, quanto maior a complexidade maior será a dificuldade de detectar um defeito. A qualidade inicial do produto se refere ao número de defeitos que podem estar contidos no artefato de software, ou seja, quanto maior for a qualidade menor será o número de defeitos existentes.

3.2 Modelo de Dinâmica de Sistemas

Após identificar os relacionamentos entre as variáveis, com base em dados extraídos da literatura, foram construídos diagramas de influência para representar as relações de causa e efeito entre as variáveis. Em seguida, esses diagramas de influência foram mapeados em um modelo de dinâmica de sistemas, utilizando-se as seguintes regras:

- Variáveis que representam algo que sofre acúmulos ou decrementos ao longo do tempo foram mapeadas em estoques;
- Variáveis que influenciam diretamente outra variável previamente mapeada em um estoque foram mapeadas em fluxos;
- Variáveis que representam os parâmetros de entrada para realizar a simulação do modelo foram convertidas em conversores constantes, ou seja, seu valor não é alterado durante toda simulação;
- Variáveis não classificadas em nenhuma categoria anterior (estoques, fluxos e constantes) foram mapeadas em conversores.

Para construir o modelo de dinâmica de sistemas foi utilizada a ferramenta Vensim [19], em sua versão gratuita para uso acadêmico. O modelo obtido é apresentado na Fig. 3.

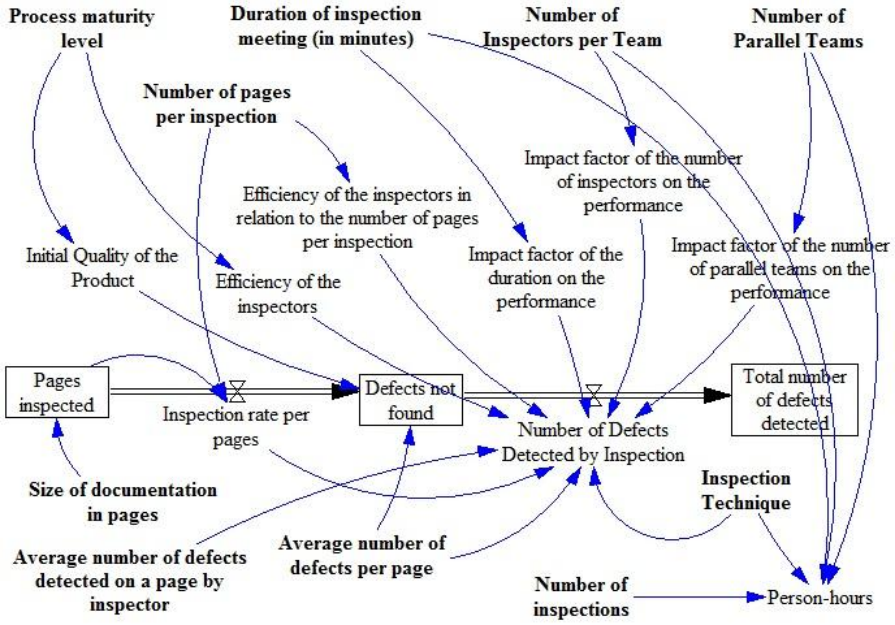

Fig. 3. Modelo estoque e fluxo da Dinâmica de Sistemas.

A variável *Process maturity level* representa o nível CMMI da empresa. Ela influencia positivamente as variáveis *Efficiency of the inspectors* e *Initial Quality of the Product*, que representam respectivamente a produtividade dos inspetores e a qualidade inicial do artefato que será inspecionado. O relacionamento de influência da variável *Process maturity level* sobre as variáveis *Efficiency of the inspectors* e *Initial Quality of the Product* foi quantificado com base nos estudos realizados por [4]. Variando o nível CMMI da empresa do mais baixo para o mais alto, a produtividade dos inspetores e a qualidade final do produto podem aumentar em até 329% e 132%, respectivamente.

A variável *Number of pages per inspection* representa o número de páginas do documento alocadas por inspeção e influencia positivamente na taxa de inspeção por páginas, representada pelo fluxo *Inspection rate per pages*, e negativamente na eficiência dos inspetores, representada pela variável *Efficiency of the inspectors in relation to the number of pages per inspection*. O relacionamento de influência da variável *Number of pages per inspection* sobre a eficiência dos inspetores foi modelada como uma variável-gráfica baseada nos experimentos realizados por [11]. Uma variável-gráfica é utilizado quando não se conhece a equação matemática que define o relacionamento entre duas variáveis e permite quantificar o relacionamento entre variáveis por meio de um esboço do gráfico correspondente à função matemática que quantifica o relacionamento.

O tempo de reunião da inspeção representado pela variável *Duration of inspection meeting* influencia não só na taxa de defeitos detectados mas também nos custos. A variável *Impact factor of the duration on the performance* é uma variável-gráfica que representa o esboço do gráfico correspondente à função matemática que quantifica o

relacionamento de influência do tempo de inspeção sobre a taxa de defeitos detectados. O esboço do gráfico foi feito com base nos estudos realizados por [5] que recomenda que o tempo de reunião da inspeção não ultrapasse 2 horas.

A variável *Number of Inspectors per Team* representa o tamanho da equipe de inspetores responsável pela atividade de detecção de defeitos. O relacionamento de influência sobre a taxa de defeitos detectados é definido pela variável-gráfica *Impact factor of the number of inspectors on the performance*. O esboço do gráfico foi feito com base nos experimentos empíricos realizados por [20], que chegou a simular o processo de inspeção com 73 inspetores.

A variável *Number of Parallel Teams* representa o número de equipes de inspetores paralelas que serão utilizadas pela técnica de leitura N-fold. O esboço do gráfico que quantifica o relacionamento de influência do número de equipes paralelas sobre a taxa de detecção de defeitos é representado pela variável-gráfica *Impact factor of the number of parallel teams on the performance*, e foi modelado baseando-se nos experimentos realizados por [10], que simulou o processo de inspeção com até 10 equipes paralelas de inspetores, aumentando o número de defeitos detectados em até 100%.

O tamanho do documento que será simulado é representado pela variável *Size of documentation in pages* que pode ter no máximo 60 mil páginas. Esse limite é baseado nos estudos de [4] que cita documentos de código-fonte com o mesmo tamanho adotado no modelo.

A variável *Number of inspections* representa o número de inspeções que serão realizadas no documento. Já a técnica de leitura utilizada nas simulações é determinada pela variável *Inspection Technique*. A variável *Person-hours* representa o esforço total necessário para que as inspeções sejam realizadas. As variáveis *Average number of defects per Page* e *Average number of defects detected on a page by inspector* representam respectivamente a média de defeitos por página e a média de defeitos detectados em uma página por um único inspetor.

O fluxo *Rate of the number of defects detected by inspection* representa o número de defeitos que são detectados por inspeção. O fluxo *Inspection rate per pages* representa o número de páginas alocadas para cada inspeção. O estoque *Total number of defects detected* representa a soma total dos defeitos detectados pelas inspeções e o estoque *Defects not found* representa o total de defeitos que não foram detectados com as inspeções. O estoque *Pages inspected* representa o total de páginas do documento que será inspecionado.

3.3 Painel de Controle

O painel de controle é uma ferramenta que permite aos usuários interagir com o modelo e realizar os ajustes para configurar os cenários a serem simulados. Para o modelo descrito neste artigo, foi definido um painel de controle, apresentado na Fig. 4, que permite aos gerentes de projetos ajustar os valores das variáveis de entrada do modelo de acordo com as características específicas da organização, da equipe e do tipo de inspeção que definem o contexto do cenário que se deseja simular.

Cada variável de entrada do modelo pode assumir valores dentro de uma faixa contínua que possui um valor mínimo e um valor máximo. Os valores das variáveis

podem ser alterados por meio de controles deslizantes (slide bars) ou manualmente. Caso o usuário informe um valor inválido para alguma variável, esta receberá automaticamente o valor mais próximo do valor informado. Ao utilizar o painel de controle, o usuário deve ter o cuidado de não informar uma combinação de valores incompatíveis para as variáveis, para que não ocorra desestabilização do modelo e não sejam gerados resultados incoerentes com a realidade. O tomador de decisão deve conhecer muito bem o problema e as combinações aceitáveis de valores para as variáveis.

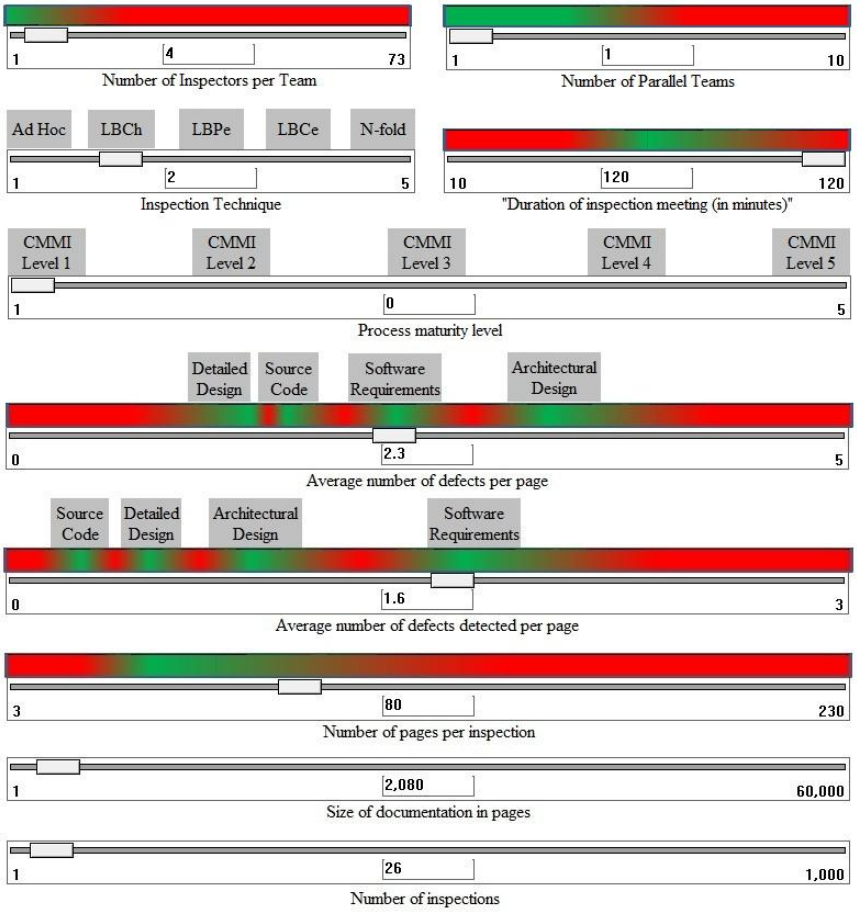

Fig. 4. Painel de controle para configurar cenários.

Algumas variáveis de entrada no painel de controle do modelo têm a sua faixa de valores permitidos representada por uma barra com a cor variando de verde a vermelho. Os valores próximos à cor verde estão dentro da faixa esperada e descrita na literatura; enquanto que os valores próximos à cor vermelha podem levar à desestabilização do modelo. Para as variáveis *Average number of defects per page* e *Average number of defects detected on a page by inspector* os valores próximos à cor verde representam os valores médios obtidos em experimentos publicados na

literatura considerando cada tipo de documento inspecionado. Já para as variáveis *Duration of inspection meeting (in minutes)*, *Number of Inspectors per Team*, *Number of pages per inspection* e *Number of Parallel Teams*, os valores próximos à cor verde são os recomendados em diversos experimentos publicados na literatura, ou seja, ao definir valores fora dessa faixa os gerentes de projetos estarão simulando cenários que representam situações reais onde recursos seriam desperdiçados na realização das inspeções. A variável *Inspection Technique* representa as 5 técnicas de leitura que podem ser simuladas: Ad Hoc, LBCh, LBPe, LBCe e N-fold. E por último, a variável *Process maturity level* representa os 5 níveis CMMI [4] que podem ser determinados nas simulações.

4 Simulações e Análises dos Resultados

As seções seguintes descrevem os resultados de simulações realizadas com o modelo de dinâmica de sistemas construído. Inicialmente foi definido e simulado um cenário base e posteriormente foram realizadas simulações de outros cenários obtidos a partir de alterações de alguns valores definidos para o cenário base. Os resultados obtidos permitiram avaliar antecipadamente os impactos que as variáveis de entrada do modelo exercem sobre o número de defeitos detectados e sobre os custos das inspeções. Os valores assumidos pelas variáveis do cenário base apresentados na Tab. 1 representam valores médios para um cenário real, e foram baseados nos experimentos de [11] e [4]. Este cenário será utilizado nas simulações apresentadas a seguir, com variações em algumas das variáveis cuja influência se deseja visualizar.

Tabela 1. Valores das variáveis utilizadas no cenário base.

Variável	Valor
Number of pages	2000
Average number of defects per Page	1.6
Average number of defects detected by inspector	0.2
Number of pages per inspection	40
Number of inspections	50
Process maturity level	CMMI Level 1
Inspection Technique	Checklist
Duration of inspection meeting	120 minutes
Number of parallel teams	1
Number of inspectors per team	4

4.1 Influência do Número de Equipes de Inspetores Paralelas

Adicionar equipes de inspetores ao processo de inspeção pode aumentar de forma significativa a taxa de detecção de defeitos, considerando que novas equipes encontrariam defeitos que não seriam detectados pelas equipes já existentes [10]. O

gráfico da Fig. 5 mostra os resultados das simulações ao considerar diferentes quantidades de equipes paralelas e permite analisar o impacto da adição de times de inspetores sobre o número de defeitos detectados e sobre os custos.

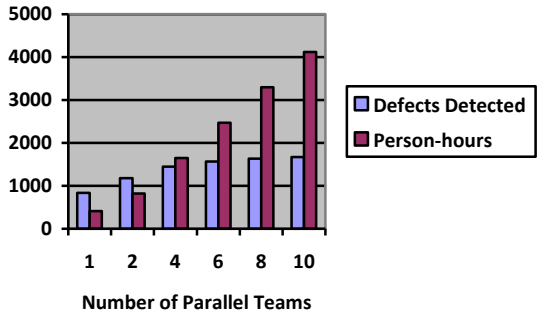

Fig. 5. Total de defeitos detectados e custo gasto em relação ao número de equipes paralelas.

O gráfico da Fig. 5 mostra que ao utilizar 1 equipe de inspetores foram detectados 833 defeitos, sendo gastos 412 homens-horas. Entretanto, ao utilizar 10 equipes paralelas foram detectados 1674 defeitos, necessitando de 4120 homens-horas. Observa-se que o aumento no número de equipes paralelas de inspetores não implica em um aumento proporcional do número de defeitos detectados. Portanto, a escolha do melhor valor para a quantidade de equipes depende da disponibilidade de recursos e do conhecimento do problema.

4.2 Influência do Tamanho da Equipe de Inspetores

Similarmente à variável *Number of Parallel Teams*, o tamanho da equipe de inspetores também influencia de forma significativa o número de defeitos encontrados e os custos. Entretanto, ao adicionar um inspetor à equipe, além de aumentar a taxa de detecção de defeitos, o tempo total gasto para inspecionar o documento diminui, já que uma parte do documento é atribuída a esse novo inspetor. Os resultados obtidos ao variar o número de inspetores podem ser observados no gráfico da Fig. 6.

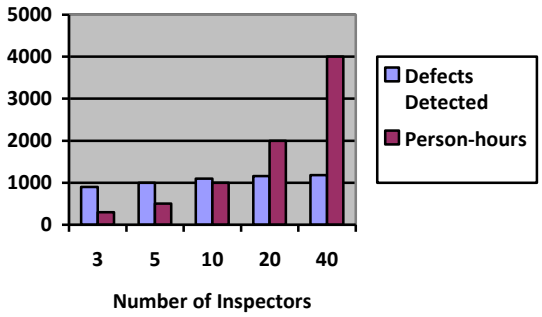

Fig. 6. Total de defeitos detectados e custo gasto em relação ao número de inspetores.

O gráfico da Fig. 6 mostra que ao utilizar 3 inspetores foram detectados 896 defeitos e foram gastos 300 homens-horas. E ao utilizar 10 inspetores, o número de defeitos detectados foi 1098 e o custo foi de 1000 homens-horas. É possível observar também que o crescimento do número de defeitos detectados ao utilizar mais de 20 inspetores é relativamente baixo, enquanto que os custos aumentam significativamente. Logo, a escolha do número de inspetores a serem utilizados na atividade de detecção de defeitos depende do total de recursos que podem ser gastos, observado o limite a partir do qual não há melhora nos resultados.

Experimentos disponíveis na literatura variam nas suas recomendações quanto ao tamanho ideal da equipe de inspetores. [14] recomenda entre 3 e 5 revisores, e para [10], o ideal deve ser 3 ou 4.

4.3 Influência da Técnica de Leitura

Segundo [3], inspeções realizadas em artefatos da fase de requisitos com o uso das técnicas de leitura baseadas em perspectiva encontrariam, em média, 35% mais defeitos que inspeções ad-hoc. Por outro lado, este mesmo autor reconhece que a aplicação da técnica requer um esforço 30% maior. Esse estudo deixa claro que cada técnica utilizada na inspeção de software possui níveis de eficiência e esforços bastante variados. Portanto, nem sempre a utilização da técnica com maiores índices de detecção de defeitos será a melhor escolha para o projeto. Os resultados obtidos ao simular o uso dos diferentes tipos de técnicas de leitura podem ser vistos analisando o gráfico da Fig. 7.

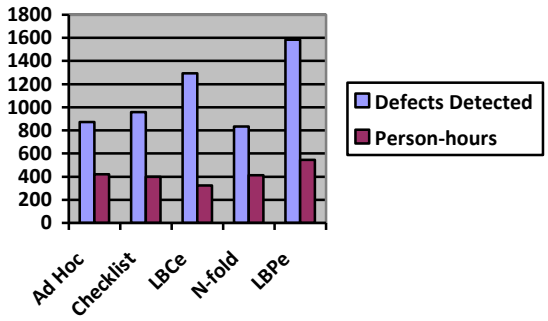

Fig. 7. Total de defeitos detectados e custo gasto em relação à técnica de leitura utilizada.

Detectando 1293 defeitos com a utilização da técnica de leitura baseada em cenários, observou-se que foram detectados 35% e 48% de defeitos a mais que as técnicas de leitura Checklist e Ad Hoc, respectivamente. Outra observação que pode ser notada ao analisar o gráfico é a superioridade da técnica LBPe em relação às demais, que segundo experimentos realizados no mundo real é a técnica mais eficiente entre todas.

5 Conclusões e Trabalhos Futuros

Ao utilizar o modelo de dinâmica de sistemas descrito neste artigo, gerentes de projeto podem realizar simulações para obter um diagnóstico "a priori" sobre os possíveis impactos da realização de inspeções nos diferentes tipos de documentos de projetos de software. O modelo permite aos gerentes definir cenários de acordo com as características da empresa, do projeto e da equipe que se deseja simular, permitindo antever os impactos das decisões gerenciais acerca da maneira como será realizada a atividade de inspeção. As decisões e intervenções gerenciais podem ser testadas com a realização de simulações e os resultados obtidos permitem analisar o retorno obtido com a alocação de recursos para a atividade de inspeção.

Em empresas que já realizam inspeções durante o desenvolvimento de software, o modelo pode ser utilizado para analisar o impacto de alterações no modo como essa atividade é realizada. As simulações com o modelo permitem reduzir o tempo, o custo e o risco dos experimentos, que não precisam ser feitos em projetos reais. Diversos cenários podem ser testados até que seja obtido o mais adequado para a empresa, em termos de número de defeitos detectados e recursos alocados.

Com o objetivo de desenvolver um modelo capaz de produzir resultados cada vez mais próximos da realidade, os próximos passos desse trabalho são: melhorar a quantificação dos relacionamentos entre as variáveis presentes no modelo e integrar o modelo descrito nesse trabalho com os modelos produzidos por [2] e [8] para obter um modelo mais completo e robusto, abordando várias atividades do processo de desenvolvimento de software.

Referências Bibliográficas

1. Abdel-Hamid, T. K. Investigating the cost/schedule trade-off in software development. IEEE Software, jan., 1990.
2. Ambrósio, B. G.; Braga, J. L.; Resende Filho, M. A. Modeling and scenario simulation for decision support in management of requirements activities in software projects. Journal of Software Maintenance and Evolution (Print), v.23, p.35-50, 2011. http://dx.doi.org/10.1002/smr.469.
3. Basili, V. R. et al. The Empirical Investigation of Perspective-Based Reading. Journal of Empirical Software Engineering, v. 1, n. 2, 1996.
4. D. Gibson; D. Goldenson; K. Kost. "Performance Results of CMMI-Based Process Improvement". Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, Technical Report CMU/SEI-2006-TR-004, 2006.
5. Dunsmore, A.; Roper, M.; Wood M. Object oriented inspection in the face of delocalisation. Proceedings of the 22nd International Conference on Software Engineering, (Limerick), 2000.
6. Fagan, M. E. Advances in Software Inspections, IEEE Transactions of Software Engineering, v. SE-12, n. 7, p. 744-751, 1986.
7. Forrester, J. W.; Senge, P. Tests for building confidence in system dynamics models. A. Legasto et al. (eds.), TIMS Studies in the Management Sciences (System Dynamics), North-Holland, The Netherlands, p. 209-228, 1980.
8. Hermsdorf, V. O. et al. Modelagem da atividade de elicitação de requisitos utilizando a técnica de entrevista: uma abordagem utilizando dinâmica de sistemas. XIV WER - Workshop em Engenharia de Requisitos Proceedings XIV CibSE. Rio de Janeiro, RJ, v. 14. p. 309-320, 2011.
9. Jung, C. F. Metodologia para Pesquisa & Desenvolvimento. Rio de Janeiro: Axcel Books, 2004.
10. Kantorowititz, E.; Guttman, A.; Arzi, L. The Performance of the N-Fold Requirement Inspection Method. Requirements Engineering, v.2., n. 3, p. 152-164, 1997.
11. Kelly, J. C.; Sheirif, J. S.; Hops, J. An analysis of defect densities found during software inspections. Journal of Systems Software, v.17, n. 2, p. 111-117, 1992.
12. Knight, J. C.; Myers, E. A. An Improved Inspection Technique. Communications of the ACM, v. 36, n. 11, p. 51-61, 1993.
13. Laitenberger, O.; Debaud, J. An encompassing life cycle centric survey of software inspection. Journal of Systems and Software, v. 50, n. 1, p. 5-31, 2000.
14. Madachy, R. J. System dynamics modeling of an inspection-based process. Proceedings of the 18th international conference on Software engineering, 1996.
15. Porter, A. A.; Votta, L. G.; Basili, V. R. Comparing detection methods for software requirements inspections: a replicated experiment. IEEE Trans. Software Engineering, v. 21, n. 6, p. 563-575, 1995.
16. Russel, G. W. Experience with Inspection in Ultra large-Scale Developments. IEEE Software, v. 8, n. 1, p. 25-31, 1991.
17. Shull, W. et al. What We Have Learned about Fighting Defects. International Software Metrics Symposium, Ottawa, Canada, 2002.
18. Sterman, J. D. (2000) “Business dynamics: systems thinking and modeling for a complex world”. Boston, MA: Irwin McGraw-Hill, 2000. 982 p.
19. Vensim (2010). Vensim from Ventana Systems, Inc. Disponível em: <http://www.vensim.com>. Acesso em: 3 de mar. de 2012.
20. Walia, G.; Carver, J. The Effect of the Number of Defects on Estimates Produced by Capture-Recapture Models. Proceedings of the 19th IEEE International Symposium on Software Reliability Engineering - ISSRE'2008, Seattle, WA, USA, p. 305-306, 2008.