RECEBIDO
18/03/2017
APROVADO
02/04/2018

Uma análise qualitativa-sistemática da relação entre o acúmulo de dívida técnica e a satisfação de usuários ao longo da operação de pacotes de software empresarial

A qualitative-systematic analysis of the relationship between technical debt accumulation and users' satisfaction throughout enterprise software packages operation

Eduardo Franco[1] - Univ. de São Paulo - Escola Politécnica - Dep. de Engenharia de Computação e Sistemas Digitais
Kechi Hirama[2] - Univ. de São Paulo - Escola Politécnica - Dep. de Engenharia de Computação e Sistemas Digitais
Rogério Rossi[3] - Univ. de São Paulo - Escola Politécnica - Dep. de Engenharia de Computação e Sistemas Digitais

RESUMO A operação e manutenção de pacotes de software empresarial representa a maior parcela dos investimentos totais em iniciativas dessa natureza, que podem se estender por anos ou décadas. Ao longo desse período, o pacote de software sofre influências diversas que limitam sua capacidade de se adaptar a cenários emergentes e torna a avaliação de cenários de longo prazo uma atividade não trivial. Este trabalho investiga a relação entre o acúmulo da dívida técnica, incorrido ao longo do ciclo de operação e manutenção, e a satisfação dos usuários desses pacotes. A metodologia utilizada para a pesquisa consiste em uma análise qualitativa-sistemática baseada na revisão da literatura e em ferramentas da Dinâmica de Sistemas. Apresenta-se uma conceituação sistêmica da inter-relação entre a dívida técnica e a satisfação dos usuários decorrente das influências causadas durante o ciclo de operação e manutenção do software baseado nessa conceituação, discutem-se as relações causais e suas consequências relativas à manutenção de pacotes de software no acúmulo da dívida técnica e na satisfação de seus usuários. Desta forma, este trabalho busca suprir lacunas de trabalhos anteriores, que trataram o contexto de custo e benefício obtidos das atividades de manutenção por meio de modelos estatísticos, e estabelecer uma discussão sistêmica sobre manutenção de pacotes de software comerciais e seus resultados de longo prazo.
Palavras-chave: Dívida técnica. Evolução de software. Satisfação dos usuários.

ABSTRACT *The operation and maintenance of enterprise software packages represents the most significant share of total investments in such initiatives, which may extend over years or decades. Throughout this period, the software package undergoes a variety of influences that limit its ability to adapt to emerging scenarios and makes the evaluation of long-term scenarios a non-trivial activity. This work investigates the relation between the technical debt accumulation, incurred during the operation and maintenance cycle, and the user satisfaction with these systems. The methodology employed consists of a qualitative-systematic analysis based on a literature review and a subset of Systems Dynamics tools. A systemic conceptualization of the interrelationship between technical debt and user satisfaction due to the influences caused during the software operation and maintenance cycle is presented. Based on this conceptualization, we discuss the casual relations and consequences of decisions relative to the maintenance of software packages in the satisfaction of its users. This work seeks to fill the gaps of previous studies, which addressed the costs and benefits obtained from maintenance activities in an aggregated way through statistical models, and to establish systemic discussion regarding the maintenance of commercial software packages and their long-term results.*
Keywords: *Technical debt. Software evolution. Users satisfaction.*

1. eduardo.franco@usp.br; 2. kechi.hirama@usp.br; 3. rogeriorossi8@gmail.com

FRANCO, E.; HIRAMA, K.; ROSSI, R. Uma análise qualitativa-sistemática da relação entre o acúmulo de dívida técnica e a satisfação de usuários ao longo da operação de pacotes de software empresarial. **GEPROS. Gestão da Produção, Operações e Sistemas**, Bauru, Ano 14, nº 4, out-dez/2018, p. 263-288.
DOI: 10.15675/gepros.v13i4.2033

1. INTRODUÇÃO

Há um crescente interesse sobre o tema da complexidade em pesquisas relacionadas à sistemas baseados em *software* (NAN, 2011; SOMMERVILLE et al., 2012; WHITNEY; DANIELS, 2013), em especial com relação a dimensão dinâmica (GEORGANTZAS; KATSAMAKAS, 2008; GERALDI; MAYLOR; WILLIAMS, 2010). Apesar do crescimento recente do interesse sobre o tema da complexidade da comunidade de gestão de sistemas de informação e engenharia de *software*, o assunto vem sendo discutido há décadas em outras áreas da ciência (WEAVER, 1948).

Stoyenko (1995) define sistemas computacionais como aqueles constituídos por múltiplos componentes (incluindo *software*, *hardware* e comunicação), que interagem com elementos externos, que operam de forma ininterrupta e adaptável, que apresentam degradação progressiva, que interagem com pessoas e que produzem reações imprevisíveis quando submetidos a sequência de eventos não esperados. Esses sistemas possuem tempo de vida longo (anos ou décadas) e durante o período de operação, seus componentes evoluem, suas interconexões lógicas e físicas se alteram e suas interfaces e semânticas operacionais mudam, ampliando a complexidade do sistema.

Atualmente, alguns sistemas computacionais fortemente baseados em *software* incorporam *softwares* complexos denominados pacotes de *software* empresarial.

O objeto de análise deste estudo consiste em pacotes de *software* empresarial (por exemplo, ERP – *Enterprise Resource Planning*), que operam em ambientes corporativos, de uso compulsório, que automatizam uma atividade humana ou social, que fazem suposições em relação ao mundo real e que interagem com ele provendo ou requisitando serviços. Em geral pacotes de *software* dessa categoria, devem ser adaptados para ajustarem-se às mudanças ocorridas em seu contexto operacional e que afetam a satisfação de seus usuários. Como esse contexto é dinâmico, esses *softwares* devem ser continuamente ajustados para permanecerem fiéis ao seu domínio e sua aplicação, compatíveis com seu ambiente de operação e relevantes aos objetivos e expectativas dos seus *stakeholders* (COOK et al., 2006).

As iniciativas de manter os *softwares* ajustados com esse objetivo podem resultar em sucesso ou fracasso devido ao próprio *software* e o ambiente onde ele opera.

Sauer (1993), ao analisar fracassos de iniciativas envolvendo sistemas de informação baseados em *software*, propôs que eles apenas podem ser considerados como um fracasso quando ocorre o cancelamento do seu desenvolvimento ou de sua operação. Com base neste critério de fracasso, os sistemas de informações baseados em *software* se aproximam da abordagem dos sistemas naturais, onde os comportamentos observados são explicados em termos dos objetivos de sobrevivência. A sobrevivência de um *software* em operação é obtida por meio de recursos diversos (financeiros, pessoas, infraestrutura etc.) que apoiam a continuidade de sua operação, sendo assim, ele não pode ser considerado um fracasso enquanto continuar atraindo aportes de recursos e gerando benefícios (YEO, 2002).

Lehman (1980) constatou que sistemas baseados em *software*, assim como sistemas naturais, evoluem como resposta às reações e pressões sofridas do ambiente externo e originadas por mudanças nos padrões operacionais, funcionais e estruturais, o que inevitavelmente torna-os mais complexos, inflexíveis e resistentes a mudanças. Para sobreviverem, esses sistemas devem manter sua adaptabilidade e capacidade de modificação, sendo que o nível que estas características são mantidas pode fazer muita diferença para avaliar seu sucesso ou fracasso, retorno ou prejuízo dos investimentos realizados.

A adaptação às mudanças operacionais e a evolução de um pacote de *software* operacional ocorre por meio do processo de manutenção. Nesse processo, os responsáveis pela sua manutenção podem violar as boas práticas estruturais relacionadas a arquitetura e a codificação. A metáfora "dívida técnica" (do inglês *technical debt*) foi cunhada por Cunningham (1993) para descrever o passivo acumulado pelas decisões, intencionais ou não, de entregar o *software* com qualidade abaixo do ideal para atingir objetivos de curto prazo para o negócio (como por exemplo, realizar entregas com prazos reduzidos).

A dívida técnica se refere ao acúmulo das violações ocasionadas por decisões tomadas no passado que aumentam o custo associado a manutenção de sistemas baseados em *software* e reduzem sua capacidade de modificação para atender às necessidades atuais e futuras do negócio (manutenibilidade). Incorrer em acumular dívida técnica pode acelerar a entrega de funcionalidades para atender às necessidades imediatas, no entanto, essas entregas também acarretam em um aumento na quantidade de defeitos e violações não resolvidas, reduz a flexibilidade e capacidade de modificação do *software*, aumenta o custo e o tempo necessários para realizar as manutenções e, consequentemente, reduz a satisfação dos usuários no longo-prazo (RAMASUBBU; KEMERER; WOODARD, 2015).

Esse contexto faz da avaliação de cenários de longo prazo envolvendo sistemas de informação baseados em *software*, operando em contextos dinamicamente complexos, uma atividade não trivial. Dessa forma, este trabalho tem por objetivo abordar a relação entre o acúmulo de dívida técnica e a satisfação dos usuários de pacotes de *software* empresarial ao longo de seu ciclo de operação e manutenção. Para atingir esse objetivo, apresenta-se uma análise qualitativa-sistemática, de caráter exploratória, para identificar os elementos envolvidos e seus inter-relacionamentos causais para em seguida construir uma hipótese dinâmica que explique endogenamente comportamentos observados a partir de dados secundários.

O trabalho está organizado em cinco seções. Além da introdução apresentada, a seção dois aborda a revisão da literatura utilizada para contextualizar o problema, identificar o estado da arte, trabalhos correlatos e elementos para endereçar o objetivo definido, a seção três apresenta a metodologia de pesquisa adotada para a execução do trabalho, a seção quatro apresenta a análise dos resultados obtidos e, por fim, a seção cinco apresenta as conclusões, limitações e sugestões de trabalhos futuros.

2. REVISÃO DA LITERATURA

Esta seção apresenta uma visão geral da literatura relacionada aos conceitos empregados neste trabalho, abrangendo: evolução de *software*, dívida técnica e avaliação de sucesso envolvendo sistemas de informação baseados em *software*.

2.2. Manutenção e evolução de *software*

A partir da década de 1970, pesquisadores começaram a investigar as possíveis causas para a demanda por investimentos constantes mesmo após a conclusão do desenvolvimento de sistemas baseados em *software* e de sua entrada em operação (BELADY; LEHMAN, 1976; WOODSIDE, 1979). Lehman (1980) identificou que do total dos investimentos realizados com *software* nos Estados Unidos, 70% dos recursos eram destinados à manutenção (que considerou como qualquer tipo de alteração realizada no *software* após o início de sua operação), número que foi posteriormente ampliado para até 80% (INCOSE, 2015).

Estas investigações, juntamente com os avanços obtidos nas últimas décadas, deram origem a novas linhas de pesquisas associadas a evolução de *software* e a consolidação das leis da evolução que descrevem abstrações dos comportamentos observados baseadas em modelos estatísticos (LEHMAN, 1980; LEHMAN; FERNÁANDEZ-RAMIL, 2006).

A Tabela 1 apresenta a redação da última revisão das oito leis propostas por Lehman (1996). Diversos trabalhos foram conduzidos no intuito de confirmar e avaliar a aplicabilidade delas em contextos diversos e, apesar de alguns autores terem demonstrado que parte delas não se aplica ao domínio do *software* livre, a aplicação para *softwares* comerciais foi ratificada por diversos pesquisadores (HERRAIZ et al., 2013).

Tabela 1 - Leis da evolução de *software*.

Lei	Descrição
1. Mudanças contínuas	Um *software* em operação deve ser continuamente adaptado, caso contrário torna-se progressivamente menos satisfatório.
2. Complexidade crescente	Conforme um *software* evolui sua complexidade aumenta, a não ser que sejam realizadas atividades para manter ou reduzi-la.
3. Autorregulação	O processo global de evolução de um *software* é autorregulado, apresentando medidas de atributos do produto e processo próximos a uma distribuição normal.
4. Conservação da estabilidade organizacional	A taxa média de efetividade da atividade global de um *software* em evolução é invariante ao longo do seu ciclo de vida.
5. Conservação da familiaridade	Durante o ciclo de vida de um *software* em evolução, o conteúdo de suas versões sucessivas é estatisticamente invariante.
6. Crescimento contínuo	O conteúdo funcional de um *software* deve ser continuamente ampliado para manter a satisfação dos usuários ao longo de seu ciclo de vida.
7. Qualidade decrescente	Ao menos que seja rigorosamente mantida e adaptada para levar em consideração as mudanças no ambiente operacional, a qualidade do *software* aparentará estar decaindo.
8. Sistema de *feedback*	O processo de evolução constitui em um sistema de *feedback*, com múltiplos níveis, malhas e agentes e que deve ser tratado como tal para poder ser modificado e melhorado com sucesso.

Fonte: Adaptado de Lehman (1996).

Ao longo da operação e evolução de um *software*, diversas intervenções de manutenção são realizadas no intuito de manter sua condição operacional e atender às demandas emergentes. A norma ISO/IEC 14764:2006 estabelece quatro categorias definidas de acordo com o propósito da intervenção, são elas: 1) corretiva, modificações reativas realizadas para corrigir problemas identificados no *software* após sua entrada em operação; 2) adaptativa, modificações realizadas para manter o *software* em operação mesmo havendo mudanças no ambiente operacional; 3) perfectiva, com o objetivo de melhorar o desempenho (ou novas funções) ou a capacidade manutenção do *software*; e 4) preventiva, que buscam identificar e corrigir defeitos latentes do *software* antes que eles se manifestem (INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, 2006).

2.2. Dívida técnica

Apesar da definição simples e intuitiva, a metáfora tem sido utilizada de forma indiscriminada para descrever qualquer tipo de impedimento, atrito e obstáculo na venda, desenvolvimento, implantação, manutenção ou evolução de sistemas baseados em *software*, o que tem enfraquecido e diluído o significado da metáfora (KRUCHTEN; NORD; OZKAYA, 2012).

A metáfora da dívida técnica é composta por um conjunto de constructos abrangentes, que auxiliam na comunicação dos custos e riscos futuros relacionados a baixa qualidade estrutural de um sistema baseado em *software* (CURTIS; SAPPIDI; SZYNKARSKI, 2012):

- Violações que deveriam ser corrigidas: violações de boas práticas arquiteturais ou de codificação, sabidas que possuem uma probabilidade não aceitável de contribuir para problemas operacionais severos (interrupções, falhas de segurança, corromper dados etc.) ou para contribuir com custo excessivo de aquisição, como esforço excessivo para implementar mudanças;
- Principal: é o custo necessário a incorrer para corrigir todas as violações presentes no produto de *software* em operação; e
- Juros: corresponde ao custo continuo associado às violações não corrigidas do *software* em operação, como por exemplo, horas a mais de manutenção necessárias para realizar as atividades e utilização ineficiente dos recursos.

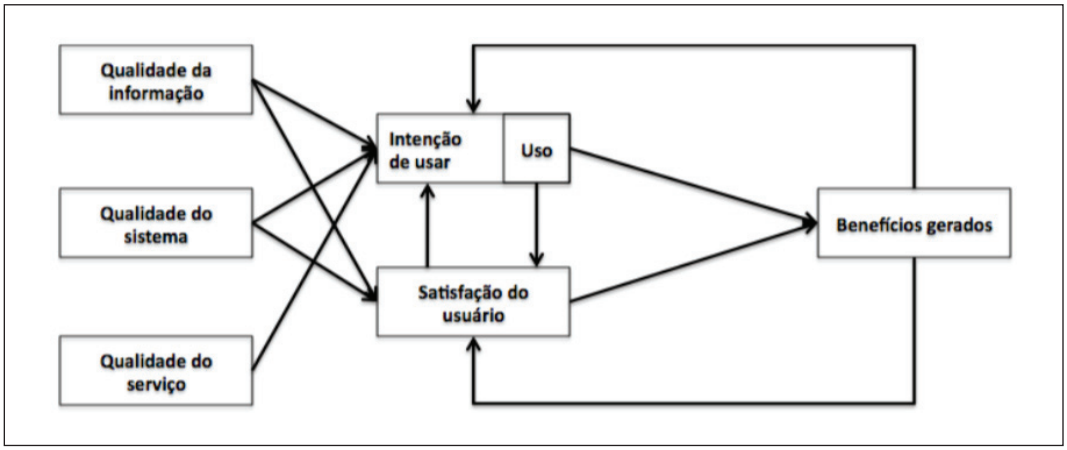
Apesar de não haver um consenso com relação ao que pode ser classificado como dívida técnica, ao nível de comprometimento dos atributos de qualidade do *software* para as violações serem classificados como tais e os limites de utilização da metáfora, um mapeamento sistemático da literatura foi conduzido para avaliar a utilização da metáfora onde os autores identificaram dez categorias: requisitos, arquitetura, projeto, código, testes, *build* (construção), documentação, infraestrutura, versionamento e defeitos (LI; AVGERIOU; LIANG, 2015).

A dívida técnica pode ser benéfica ou prejudicial para a gestão da operação e manutenção de pacotes de *software* empresarial. As dívidas que são intencionalmente incorridas para obter benefícios de curto prazo, podem ser positivas caso os custos associados sejam mantidos visíveis e sob controle (ALLMAN, 2012). Por outro lado, eles podem ocorrer de forma não intencional e não serem percebidos pelos envolvidos. Caso mantenham-se invisíveis e não resolvidos, podem se acumular e representar riscos para as atividades de manutenção e evolução a longo prazo (LI; AVGERIOU; LIANG, 2015).

2.3. Avaliação de sucesso

Dentre os diversos modelos de avaliação de iniciativas envolvendo sistemas de informação baseados em *software*, o modelo proposto por DeLone e McLean (2003) ganhou destaque e tornou-se uma das contribuições seminais para avaliar o sucesso dessas iniciativas. O modelo, apresentado graficamente na Erro! Fonte de referência não encontrada., propõe uma abordagem centrada no usuário para avaliar o sucesso, que é definida como uma variável dependente de seis dimensões interdependentes (qualidade do sistema, qualidade da informação, qualidade do serviço, intenção de utilizar e utilização do sistema, satisfação do usuário e benefícios gerados).

Figura 1 - Modelo de avaliação de sucesso de sistemas de informação.



Fonte: Adaptado de DeLone e McLean (2003).

Esse modelo foi selecionado porque ele representa uma visão processual e causal da interdependência das seis dimensões de sucesso. O foco no relacionamento causal é também um dos elementos centrais da dinâmica de sistemas, que procura explicar endogenamente comportamentos complexos for meio de interações (*feedbacks*) entre os componentes do sistema (STERMAN, 2000). Um modelo processual sugere que um sistema de informação é primeiramente criado, contendo um conjunto de funcionalidades que podem exibir diferentes níveis de qualidade do sistema e da informação nele contida. Em seguida, seus usuários ao utilizarem o sistema e suas funcionalidades, ficam satisfeitos ou insatisfeitos com o mesmo ou com as informações geradas por ele. Por fim, o impacto da experiência de cada usuário trabalhando em conjunto com os demais usuários gera os benefícios para a organização. Em contrapartida, o modelo causal avalia a covariância das dimensões de sucesso para determinar os relacionamentos causais entre elas. Por exemplo, em um sistema considerado com elevada qualidade é esperado que gere elevada satisfação dos usuários e que o sistema seja utilizado com maior frequência, consequentemente gerando ganhos de produtividade individuais que somados levem ao ganho de produtividade da organização como um todo.

Devido à má gestão da operação e manutenção, o que se inicia como uma implementação de sucesso de um pacote de *software* empresarial, pode terminar como um grande fracasso devido à influência das mudanças constantes

(primeira lei da evolução de *software*), crescimento contínuo (sexta lei da evolução de *software*) e qualidade decrescente (sétima lei da evolução de *software*) na redução da satisfação dos usuários.

Neste trabalho é considerada a influência das dimensões da qualidade do sistema (que se refere as características desejadas do sistema, como por exemplo, facilidade de uso, flexibilidade, funcionalidades etc.) e da qualidade da informação (que se refere as características da saída do sistema, relatórios gerenciais, como por exemplo, completude, exatidão, relevância etc.) no uso do sistema e na satisfação dos usuários. Esta simplificação foi adotada uma vez que a variável de interesse, relacionada ao acúmulo da dívida técnica, influencia e é influenciada diretamente por estas dimensões.

Do ponto de vista da satisfação do usuário, as dimensões da qualidade do sistema e qualidade da informação também são suficientes para descrever as relações diretas. De acordo com o "Modelo de Aceitação da Tecnologia" (TAM, do inglês *Technology Acceptance Model*) (DAVIS, 1989), a aceitação de uma tecnologia, neste caso um pacote de *software* empresarial, por parte dos usuários apenas pode ser influenciada por variáveis externas de forma indireta, influenciando no primeiro momento crenças, que são definidas em torno da facilidade de uso percebida (qualidade do sistema) e utilidade percebida (qualidade da informação) por parte dos usuários. Este modelo constitui uma ferramenta capaz de prever a aceitação de pacotes de *software* empresarial por parte dos usuários e guiar as intervenções gerenciais na redução do problema de tecnologias subutilizadas (DAVIS; BAGOZZI; WARSHAW, 1989).

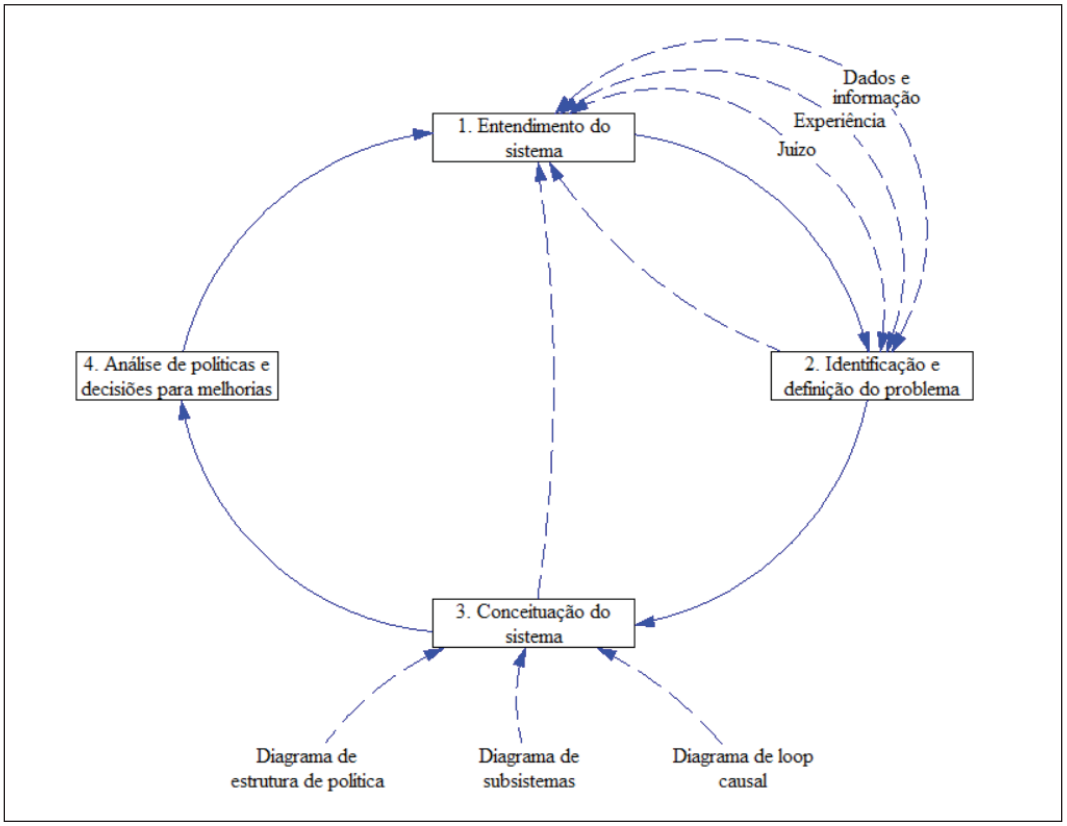
3. METODOLOGIA

A abordagem metodológica utilizada foi adaptada do trabalho de Aslani e Naaranoja (2015), onde os autores apresentaram um *framework* qualitativo-sistemático para analisar e comunicar uma visão geral do objeto de pesquisa. Este *framework* é composto pela junção de uma revisão sistemática, que consiste em uma metodologia científica empregada para identificar, avaliar e sintetizar os resultados de pesquisas primárias para apoiar a formulação de políticas, com a pesquisa qualitativa, que objetiva compreender os fenômenos e comportamentos e explicitar as razões que governam as manifestações dos mesmos. Dessa maneira, Aslani e Naaranoja (2015) classificaram esse *framework* proposto como uma pesquisa de natureza qualitativa enquadrada em uma revisão sistemática.

Abordagens similares foram aplicadas por outros trabalhos em contexto de pacotes de *software* empresarial, por exemplo, em investigações da inter-relação causal entre fatores críticos de sucesso em projetos envolvendo a implantação e uso de *Enterprise Resource Planning* (ERP), (AKKERMANS; VAN HELDEN, 2002; KING; BURGESS, 2006).

A análise qualitativa-sistemática busca entender as estruturas causais de um determinado problema antes de propor qualquer proposta de melhoria ou intervenção e, desta forma, ela analisa o problema de diferentes pontos de vista. A Figura 2 apresenta o fluxo das atividades que foram aplicadas no presente estudo e compreendem: 1) o entendimento do sistema; 2) a identificação e definição do problema; 3) a conceituação sistêmica do problema; e 4) a análise de políticas de intervenção e melhorias para o problema.

Figura 2 - Abordagem qualitativa-sistemática utilizada.



Fonte: Adaptado de Aslani e Naaranoja (2015).

A primeira etapa, referente ao entendimento sistêmico do problema, foi discutida e apresentada nas duas primeiras seções deste trabalho. As seções a seguir conceituam as etapas seguintes antes da apresentação e discussão dos resultados.

3.1. Identificação e articulação do problema

Um modelo para ser útil precisa endereçar um problema específico e precisa simplificar ao invés de reproduzir o sistema completo em detalhes. Ele deve simplificar a realidade para permitir a compreensão do contexto e do problema em investigação. A articulação do problema estabelece o propósito do modelo, que serve como critério para definir os elementos que devem ser incluídos e aqueles que podem ser ignorados para que o propósito do modelo seja atingido (STERMAN, 2000). Dois dos processos mais utilizados para este propósito, e também adotados no presente trabalho, são:

Modos de referência: onde os problemas são caracterizados dinamicamente, isto é, como os padrões de comportamento se desdobram ao longo do tempo, demonstrando como ele se originou (passado) e como ele pode evoluir ao longo do tempo (futuro).

Horizonte temporal: onde se deve retroceder no tempo até ser capaz de identificar como o problema emergiu e descrever seus sintomas. Ao mesmo tempo, deve se estender no futuro para capturar os efeitos indiretos e retardados de políticas potenciais de intervenção.

3.2. Conceituação do sistema

Aslani e Naaranoja (2015) consideram a etapa de conceituação do sistema, que compreende o componente qualitativo da abordagem empregada, o elemento central da pesquisa qualitativa-sistemática, e inclui as atividades de definição de fronteiras do problema, a identificação dos relacionamentos causais dos elementos e a discussão de políticas para intervenção no problema. Esta etapa é baseada na utilização de ferramentas oriundas da Dinâmica de Sistemas, metodologia que foi elaborada no final da década de 50 por Jay Forrester (1961).

Essas ferramentas apoiam a construção de uma hipótese dinâmica que corresponde à uma teoria em desenvolvimento que busca explicar o comportamento problemático observado, descrevendo como ele se originou e serve para guiar a modelagem das estruturas envolvidas. O foco é construir uma teoria que justifique endogenamente e que reproduza a dinâmica observada por meio da interação das variáveis e agentes representados no modelo (STERMAN, 2000).

Neste trabalho são utilizadas três ferramentas oriundas da Dinâmica de Sistemas, que são:

- Carta de limites: sumariza os conceitos que foram incluídos na análise de forma endógena (afeta o modelo e são afetadas por ele) ou exógena (afeta o modelo, mas não são afetadas por ele) e aqueles que foram excluídos (aspectos relevantes, mas que não fazem parte do foco do modelo). Ao listar elementos que foram excluídos da análise, as limitações e ressalvas dos resultados são declaradas de maneira explícita.
- Diagrama de subsistemas: apresenta a arquitetura geral do modelo constituída pelos principais subsistemas. Cada subsistema é apresentado em conjunto com os fluxos de recursos, informações e requisições que acoplam os subsistemas. Eles transmitem informações sobre a fronteira e o nível de agregação do modelo, apresentando o número e os tipos de organizações, os diferentes agentes representados e estabelecem um canal de comunicação entre o modelo mental e formal (MORECROFT, 1982).
- Diagrama causal: corresponde ao mapeamento das relações causais entre as variáveis, identificando como elas se afetam mutualmente e constituem em uma importante ferramenta para representar malhas de retroalimentação (*feedback*) de sistemas de qualquer domínio. O diagrama consiste em nós (variáveis) e seus relacionamentos (setas), onde os relacionamentos podem ser positivos ou negativos (indicado pelo símbolo correspondente na extremidade da seta).

4. RESULTADOS E DISCUSSÕES

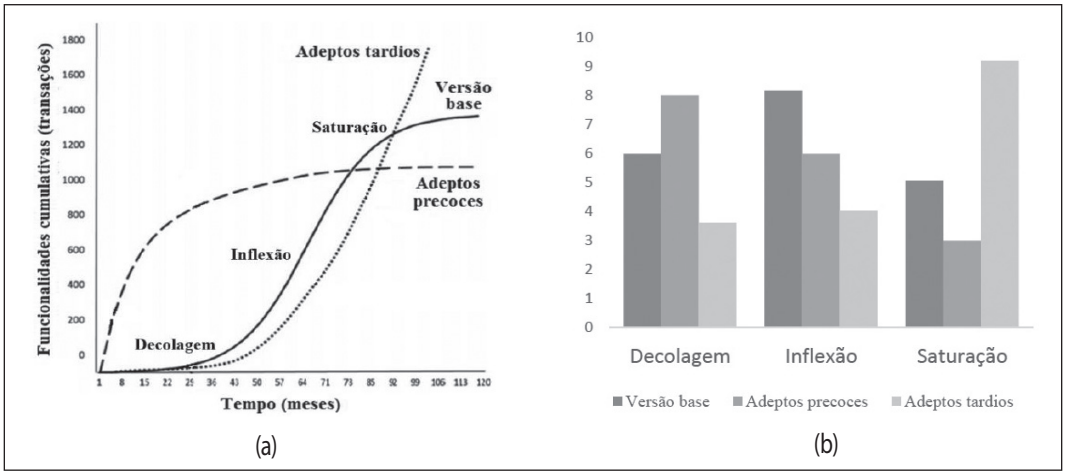
A seção a seguir apresenta os resultados obtidos do estudo, juntamente com a discussão de como estes resultados endereçam o objetivo apresentado na primeira seção.

4.1. Identificação e definição do problema

Para caracterizar o modo de referência do problema de interesse, foram utilizados dados de uma pesquisa primária identificada na revisão da literatura, onde foi conduzido um estudo longitudinal e levantados dados de empresas que adotaram um pacote de *software* empresarial modificável do tipo ERP (RAMASUBBU; KEMERER, 2014). Nesse estudo, os autores investigaram os benefícios e custos associados ao acúmulo da dívida técnica ao longo do seu tempo de vida, em termos da qualidade do sistema e da satisfação de seus usuários.

Após analisar o crescimento de funcionalidades disponíveis nos pacotes de *software* instalados ao longo de 120 meses de operação e manutenção, Ramasubbu e Kemerer (2014) identificaram três padrões evolutivos: 1) versão base, os clientes seguiram a curva de crescimento conforme a disponibilização de novas funcionalidades pelo fornecedor; 2) adeptos precoces (*early adopters*), os clientes adicionaram um conjunto funcionalidades de forma mais rápida que o disponibilizado pelo fornecedor; e os 3) adeptos tardios (*late adopters*), grupo de clientes que absorveram um conjunto menor de funcionalidades do que disponibilizado pelo fornecedor, evitando funcionalidades voláteis iniciais ("beta") e modificações customizadas para minimizar potenciais conflitos com atualizações futuras. Esses padrões evolutivos são ilustrados na Figura 3 (a).

Figura 3 - Perfis de crescimento de funcionalidades ao longo do tempo e influência na satisfação de usuários.



Fonte: Adaptado de Ramasubbu e Kemerer (2014).

A partir da taxa de crescimento das funcionalidades disponibilizadas, três fases distintas do ciclo de vida do pacote de *software* foram identificadas: 1) decolagem, onde o crescimento começa a acelerar após um crescimento inicial lento; 2) inflexão, que representa o momento onde foi atingida a metade funcionalidades totais disponíveis ao final da vida útil do *software*; e 3) saturação, onde o crescimento de funcionalidades começa a desacelerar. Para cada uma dessas três fases, foi avaliada a satisfação dos usuários em uma escala de 1-10 e o resultado apresentado na Figura 3(b).

Os adeptos precoces, apesar de apresentarem inicialmente um nível de satisfação maior dos seus usuários, atingiu um ponto de saturação de funcionalidades de forma prematura, o que reflete a menor avaliação da satisfação dos usuários no longo prazo. Em contrapartida, os clientes classificados como adeptos tardios, que apesar de um crescimento lento de funcionalidades no estágio inicial da adoção do sistema (fase de decolagem) e com isso tiveram a menor avaliação de satisfação dos usuários, apresentaram um crescimento tardio mais acelerado, em virtude da manutenção de características como flexibilidade e capacidade de modificação (baixo acúmulo de dívida técnica), e com isso obtiveram a maior avaliação da satisfação dos usuários no longo prazo e conseguiram atingir um número maior de funcionalidades em operação.

Os gráficos apresentados na Figura 3, contendo as séries temporais dos três perfis de crescimento de funcionalidades e a satisfação dos usuários associadas as fases de evolução do pacote de *software* empresarial (decolagem, inflexão e saturação), representam o modo de referência que norteou a conceituação do sistema descrita na subseção a seguir. O horizonte de tempo utilizado refere-se ao intervalo de tempo dos dados disponíveis, que é de 120 meses.

4.2. Conceituação do sistema

Esta seção apresenta a conceituação sistêmica do problema, identificando os diferentes fatores que influenciam as variáveis dependentes de interesse (satisfação dos usuários e acúmulo da dívida técnica), descrevendo endogenamente por meio das relações causais como emerge o comportamento identificado no modo de referência.

4.2.1. Carta de limites

A carta de limites, apresentada na Tabela 2, é composta por três colunas que representam as variáveis endógenas, exógenas e excluídas. Essa carta não tem o propósito de ser exaustiva, mas sim transmitir em alto nível as principais considerações dos limites estabelecidos para a análise e a conceituação proposta. Ela foi construída com base nos dados e informações analisados, na experiência e no juízo dos autores (conforme apresentado na Figura 2) para nortear o trabalho proposto e definir o escopo da análise.

Tabela 2 - Carta de limites.

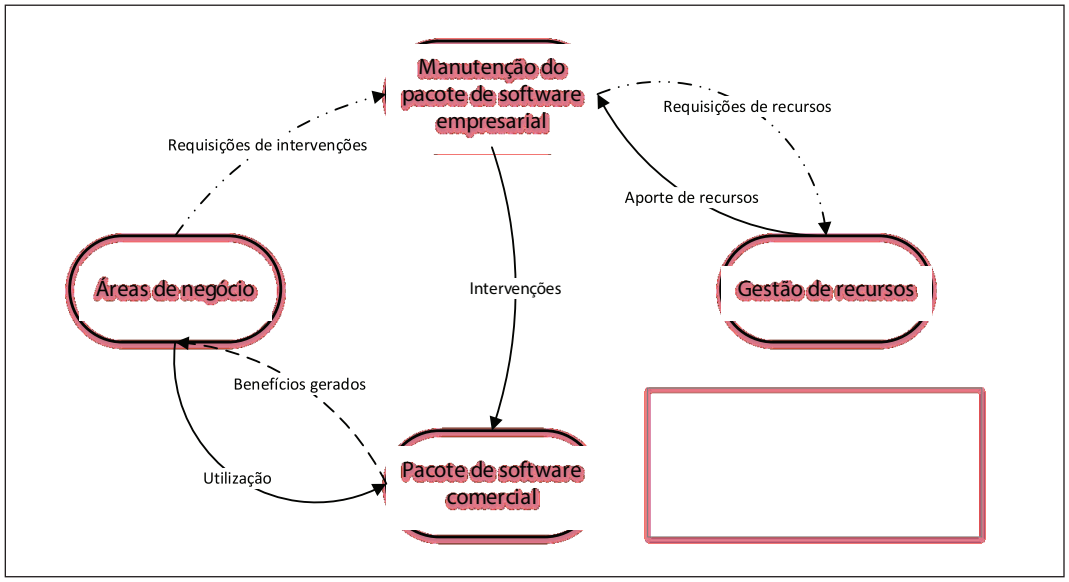
Endógenas	Exógenas	Excluídas
Backlog de atividades	Obsolescência do sistema	Infraestrutura
Funcionalidades disponíveis	Mudanças no contexto operacional	Serviços de apoio (treinamento, suporte etc.)
Dívida técnica	Recursos disponíveis	
Nível de satisfação dos usuários	Taxa de disponibilidade de novas funcionalidades pelo fornecedor	
Manutenção corretiva e adaptativa		
Manutenção perfectiva e preventiva		
Total de recursos investidos		

Fonte: Os próprios autores.

4.2.2. Diagrama de subsistemas

A Figura 4 apresenta o diagrama identificando os subsistemas que compõem o modelo, que representam: manutenção do pacote de *software* empresarial, pacote de *software* empresarial, áreas de negócio e gestão de recursos. Neste diagrama também são apresentadas as principais interações entre os subsistemas, que representam os fluxos de serviços, informações e requisições.

Figura 4 - Diagrama de subsistemas.



Fonte: Os próprios autores.

A Tabela 3 apresenta uma descrição sucinta do papel que cada subsistema representa dentro do modelo construído e as interações entre eles.

Tabela 3 - Descrição dos subsistemas do modelo.

Nome do subsistema	Descrição
Pacote de *software* empresarial	Representa o pacote de *software* empresarial, contendo características relacionadas a requisitos funcionais e não-funcionais, que interage com o ambiente externo (áreas de negócio e contexto operacional) e, ao longo do seu ciclo de vida, sofre influências descritas nas leis da evolução de *software*, como por exemplo, mudanças constantes, qualidade decrescente, complexidade crescente e crescimento continuo. No modelo são caracterizados por um conjunto de funcionalidades disponíveis e pelo acúmulo de dívida técnica incorrida devido às violações ocorridas (intencionais ou não) ao longo das manutenções realizadas.
Áreas de negócio	Corresponde às diversas áreas de negócio de uma determinada organização que interagem com o *software* por meio de seus usuários. A partir das funcionalidades disponíveis, formulam suas percepções relacionadas a à facilidade de uso percebida e à utilidade percebida, utilizam o sistema e estabelecem o nível de satisfação. A partir das funcionalidades disponibilizadas e qualidade do sistema, demandam intervenções relacionadas à manutenção corretiva e adaptativa.

Manutenção do pacote de *software* empresarial	Inclui os elementos relacionados às atividades inerentes a manutenção do *software*. Além dos recursos necessários para conduzir as demandas (representada por um *backlog* de atividades) por intervenções das áreas de negócio (manutenções adaptativas e corretivas), também corresponde às demandas de intervenções de manutenções preventivas e perfectivas decorrentes do acúmulo de dívida técnica do *software*.
Gestão de recursos	Representa o aporte de recursos (financeiro, pessoal etc.) ao subsistema de operação e manutenção do pacote de *software* para conduzir a demanda de intervenção por manutenção corretiva, adaptativa, preventiva e perfectiva. A disponibilidade de recursos é representada como um bem finito e que impõem restrições quanto capacidade de executar as intervenções demandadas e constituem uma análise de *trade-off* entre reduzir a dívida técnica para manter a capacidade de modificação do *software* ou atender às demandas dos usuários para manter ou aumentar sua satisfação.

Fonte: Os próprios autores.

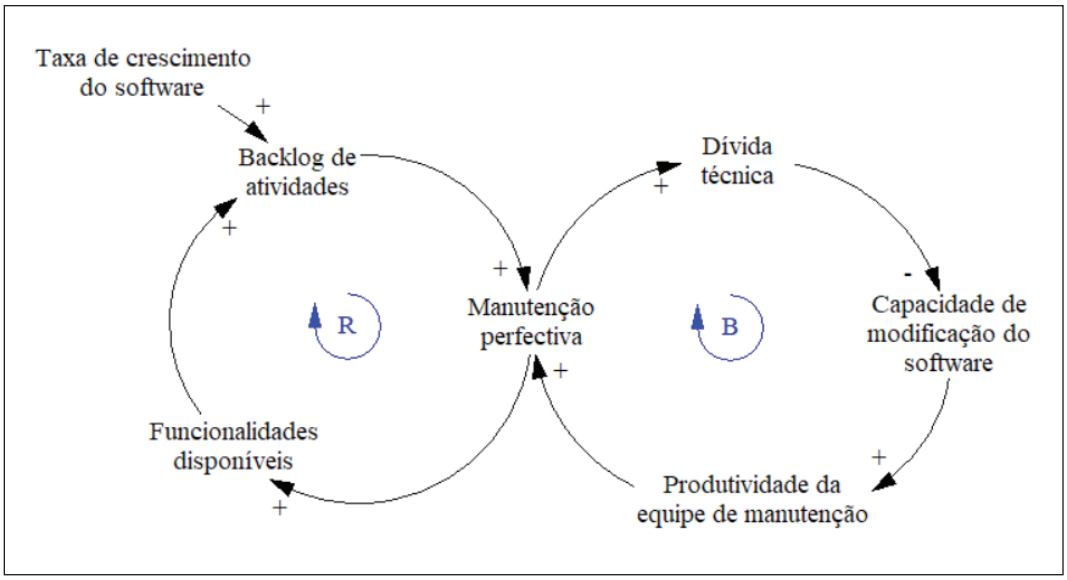
4.2.3. Diagrama causal

Os pressupostos e proposições utilizados para construção dos diagramas apresentados nesta subseção são embasados em elementos identificados por meio da revisão da literatura, apresentada na seção "2. REVISÃO DA LITERATURA", e na identificação e definição do problema, em especial no modo de referência definido (seção 3.1).

O comportamento dinâmico do crescimento das funcionalidades seguindo uma curva "S", observado na Figura 3 (a) do perfil "versão base", é capturado pelo arquétipo sistêmico do "limite do crescimento" (SENGE, 2006). Os arquétipos são úteis para construir uma visão geral da natureza do problema subjacente e oferecem uma estrutura básica na qual um modelo pode ser desenvolvido e construído. No caso do arquétipo "limite do crescimento", ele representa as causas de um sintoma onde um processo se retroalimenta e produz um período de crescimento e expansão. Em seguida, esse crescimento desacelera e eventualmente chega a ser interrompido, podendo eventualmente entrar em um colapso acelerado.

A estrutura desse arquétipo adaptado ao problema investigado no presente estudo é apresentado na Figura 5. Os sentidos dos relacionamentos são indicados pelos sinais de positivo ("+") e negativo ("-") na extremidade de cada seta. A polaridade de cada malha de relacionamento fechada (*feedback*) é identificada pelas letras "R" e "B", representando respectivamente ciclos de reforço e balanceamento da perturbação sofrida.

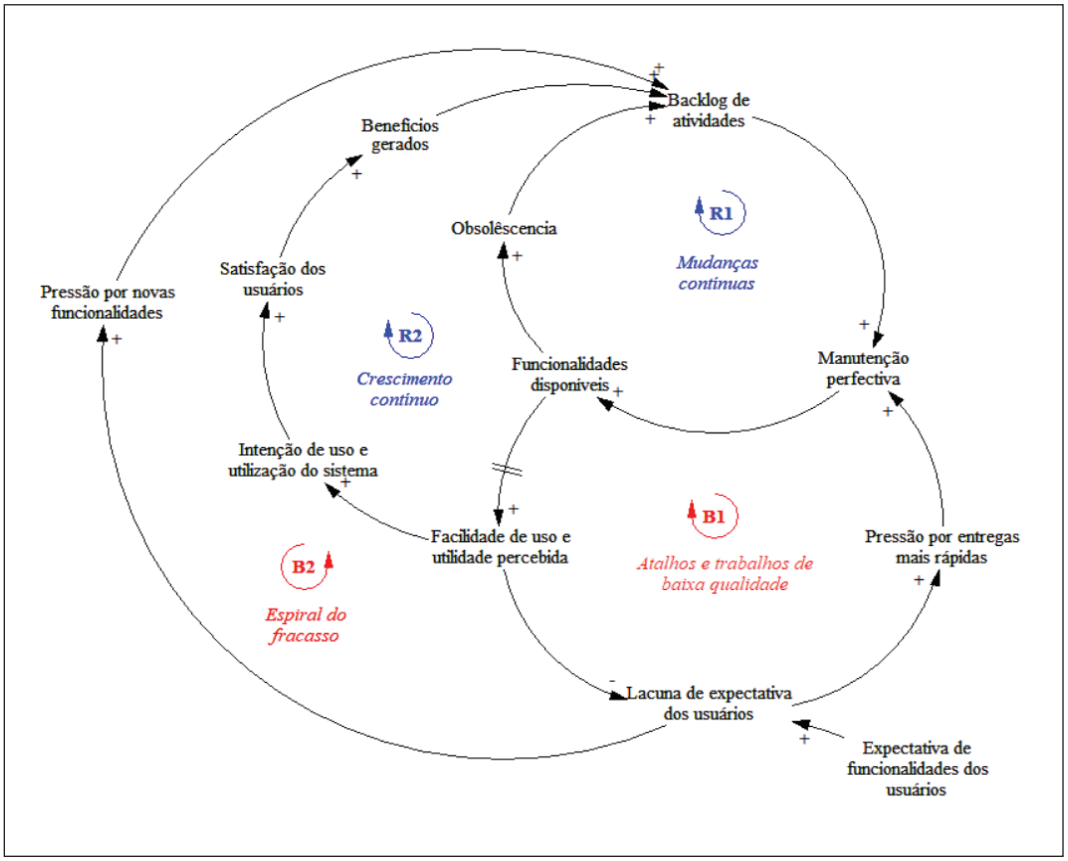
Figura 5 - Arquétipo do comportamento observado no modo de referência.



Fonte: Os próprios autores.

A malha de reforço é responsável pelo crescimento observado na fase de "decolagem" e a malha de balanceamento pelo comportamento observado na fase de "saturação". Cada uma dessas malhas exerce uma dominância em períodos distintos do ciclo de vida, e o ponto crítico da transferência dessa dominância ocorre na fase de "inflexão", onde o acúmulo da dívida técnica torna demasiadamente oneroso o crescimento das funcionalidades variáveis e suas inter-relações que atuam diretamente na evolução do pacote de *software* empresarial e na satisfação dos usuários, estão apresentadas no diagrama causal ilustrado na Figura 6Erro! Fonte de referência não encontrada..

Figura 6 - Diagrama causal relacionado à satisfação dos usuários.



Fonte: Os próprios autores.

A primeira malha de reforço, identificado por "R1 – Mudanças contínuas", representa as funcionalidades disponíveis do pacote de *software* em operação que, com o passar o tempo, sofrem a influência de mudança no contexto operacional e a descoberta de defeitos latentes, sofrendo o efeito descrito como obsolescência. Estas deficiências ao serem identificadas pelos usuários, são adicionadas ao *backlog* de atividades dos responsáveis pela manutenção corretiva e adaptativa do *software*. Esta malha corresponde a um ciclo de reforço, uma vez que quanto maior o número de funcionalidades, maior a obsolescência do *software* e, por consequência, maior será o esforço dispendido nestas manutenções caracterizando um crescimento vicioso (representando a primeira lei da evolução de *software* denominada "mudanças constantes").

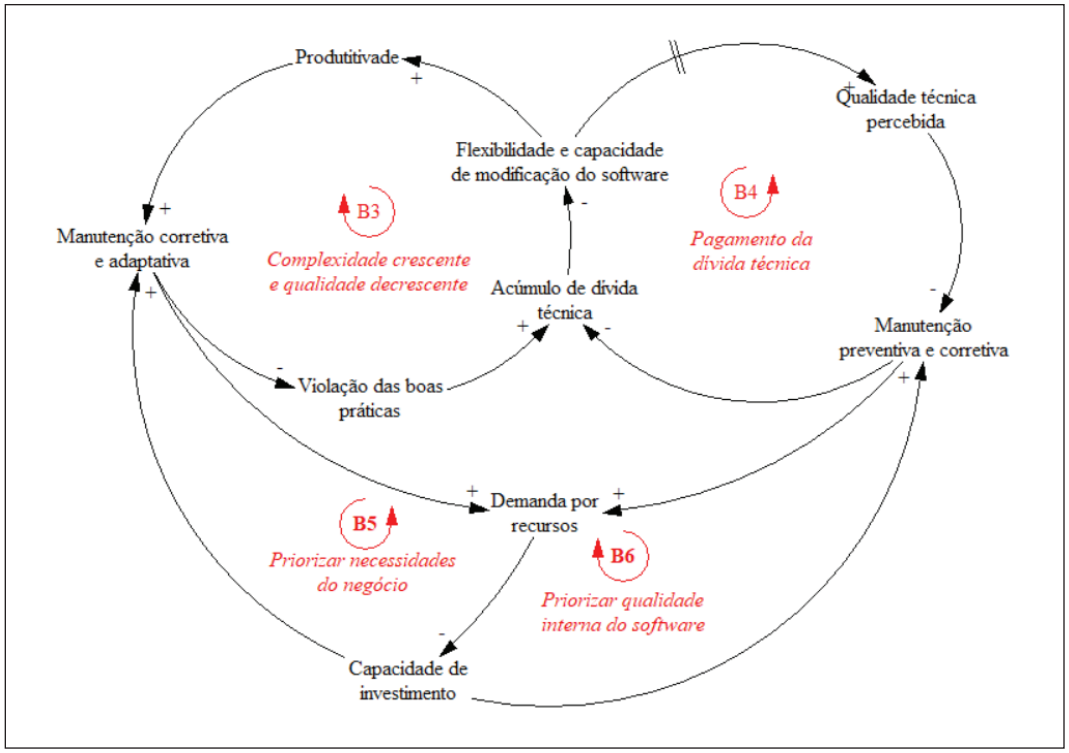
A segunda malha de reforço, identificado por "R2 – Crescimento contínuo", representa a sexta lei da evolução de *software*. Nessa malha também estão presentes as dimensões de avaliação de sucesso do modelo proposto por DeLone e McLean (2003), juntamente com as avaliações de percepção de facilidade de uso e utilidade do modelo TAM (DAVIS, 1989), que influenciam a satisfação dos usuários, os benefícios gerados e a demanda constante para o crescimento do escopo de atuação do pacote de *software* instalado para atender os objetivos do negócio.

A malha de balanceamento identificada por "B1 – Atalhos e trabalhos de baixa qualidade", atua para diminuir a lacuna de expectativa dos usuários acelerando as entregas e reduzindo o tempo de trabalho para o desenvolvimento e implementação de novas funcionalidades e, consequentemente, a qualidade das entregas. Essa malha consiste em uma das influências responsáveis pelo crescimento acelerado de funcionalidades identificadas pelo perfil evolutivo dos adeptos precoces, onde devido à uma lacuna representativa de funcionalidades disponíveis, ocorre uma pressão por novas funcionalidades, que retorna como novas demandas para manutenções perfectivas, fechando o ciclo e atuando para reduzir a lacuna.

A segunda malha de balanceamento "B2 – Espiral do fracasso" atua para atenuar a lacuna das expectativas dos usuários, onde é gerada uma pressão por novas funcionalidades e que faz aumentar o *backlog* de atividades. Como a percepção da facilidade de uso e utilidade por parte dos usuários ocorre com atraso com relação à disponibilização das funcionalidades, o volume de trabalho para a equipe de manutenção cresce mais rápido que a satisfação dos usuários o que tende a criar um gargalo para atender futuras demandas, aumentando o tempo de espera para novas entregas e pode gerar insatisfações.

A Figura 7 apresenta as variáveis e inter-relações associadas diretamente com o acúmulo de dívida técnica incorrida ao longo do ciclo de operação e manutenção. As malhas de reforço "R1 – Mudanças contínuas" e "R2 – Crescimento contínuo", identificadas na Figura 6, são restringidas pelas malhas de balanceamento "B3 – Complexidade crescente e qualidade decrescente" (reduz a flexibilidade e capacidade de mudança do *software* por meio do acúmulo da dívida técnica) e "B4 – Pagamento da dívida técnica" (compete por recursos com o desenvolvimento de novas funcionalidades). A malha "B3" representa a segunda e a sétima leis da evolução de *software*, que demandam manutenções preventivas e corretivas constantes para manter a flexibilidade e capacidade de modificação do *software*.

Figura 7 - Diagrama causal relacionado ao acúmulo de dívida técnica.



Fonte: Os próprios autores.

As diferenças entre os perfis de crescimento de funcionalidades identificados anteriormente (adeptos precoces e tardios) representam diferenças nas dominâncias entre as malhas de reforço e balanceamentos. No perfil adepto precoce, há uma dominância das malhas de "R2" e "B1", que fazem crescer as funcionalidades do pacote de *software* instalado rapidamente e com o passar o tempo, e com o acúmulo da dívida técnica e erosão da flexibilidade e capacidade de modificação do *software*, a dominância passa a ser da malha "B3" que atua para restringir as mudanças para atender às novas demandas. No perfil dos adeptos tardios, não ocorre a corrida inicial para entrega de funcionalidades, e nas fases de "inflexão" e "saturação" a malha "B3" não exerce a dominância nas malhas identificadas para atender às demandas dos usuários, uma vez que não ocorreu o crescimento descontrolado do acúmulo da dívida técnica, conseguindo manter ou até aumentar a satisfação dos usuários.

As malhas de balanceamento identificados por "B5 – Priorizar necessidades do negócio" e "B6 – Priorizar qualidade interna do *software*" representam a disponibilidade de recursos como um elemento finito e que demanda uma análise de compromisso entre investir em manutenções corretiva e adaptativa, para atender às demandas das áreas de negócio e manter a satisfação dos usuários, e em manutenções preventiva e perfectiva, para reduzir o acúmulo de dívida técnica e manter a flexibilidade e capacidade de adaptação do pacote de *software* em operação.

5. CONCLUSÃO

A análise exploratória, de natureza qualitativa-sistemática, favorece a compreensão da relação entre o acúmulo da dívida técnica, incorrido ao longo da operação e manutenção de pacotes de *softwares* empresariais, e a satisfação de seus usuários. As variáveis e seus inter-relacionamentos apresentados e discutidos na seção "4.2.3 Diagrama causal", levaram em consideração como ponto de partida a análise de dados secundários, teorias relevantes identificadas em uma revisão de literatura e na experiência e julgamento dos autores.

A conceituação do processo de manutenção e sua influência na relação entre dívida técnica e a satisfação dos usuários tomando como ponto de partida o arquétipo "limite do crescimento", levou em consideração que a fase do crescimento do pacote de *software* empresarial (que pode ter diferentes taxas de crescimento definidas pela estratégia de investimento e crescimento adotada), enfrenta em determinado momento do seu tempo de vida uma desaceleração crescente causada pela aproximação do limite exercido pela malha de balanceamento. Este limite ocorre quando o *software* atinge um nível de inflexibilidade onde a produtividade do processo de manutenção foi exaurida e eventuais novas alterações e novos acréscimos tornam-se economicamente inviáveis.

As malhas fechadas de reforço e balanceamento identificadas nos diagramas causais, explicam diferentes comportamentos observados (denominado comportamento de referência) de acordo com as dominâncias exercidas pelas malhas em cada um dos perfis de crescimento de funcionalidades em diferentes fases da evolução do pacote de *software*. A hipótese dinâmica apresentada, con-

tendo a carta de limites, o diagrama de subsistemas e o diagrama causal, pode ser utilizada como ponto de partida para a formulação de um modelo de simulação completo (contendo para ser utilizado na definição e avaliação de políticas de implantação, operação e manutenção de pacotes de *software* empresarial.

Este trabalho, ao utilizar ferramentas da Dinâmica de Sistemas, buscou suprir eventuais limitações de trabalhos anteriores que analisaram as causas e os efeitos do problema do custo crescente acarretado pela manutenção de *software* de forma agregada (por exemplo, regressões e outros modelos estatísticos), limitando a avaliação a um número reduzido de elementos que influenciam o fenômeno observado.

Para avaliar a aplicabilidade e utilidade da conceituação sistêmica apresentada neste trabalho, é necessário avançar no processo de modelagem para construir um modelo de simulação completo (utilizando diagramas de nível e taxa, equações dinâmicas e *softwares* específicos). Após a construção e validação de um modelo de simulação, ele poderá ser utilizado para obter dados quantitativos para avaliar e determinar níveis de acúmulo de dívida técnica que podem ser incorridos e ainda revertidos ao longo do tempo de vida do pacote de *software*, avaliar recursos necessários para alocar nos diferentes tipos de atividades de manutenção para manter níveis adequados da satisfação do usuário e da manutenibilidade do sistema, oferecer suporte a tomada de decisão de substituição de um sistema em operação (por exemplo, quando a manutenção torna-se mais onerosa e os benefícios menores que uma substituição por um novo sistema).

REFERÊNCIAS

AKKERMANS, H.; VAN HELDEN, K. Vicious and virtuous cycles in ERP implementation: a case study of interrelations between critical success factors. **European Journal of Information Systems**, v. 11, n. 1, p. 35-46, 2002.

ALLMAN, E. Managing Technical Debt. **Queue**, v. 10, n. 3, p. 10, 2012.

ASLANI, A.; NAARANOJA, M. A systematic-qualitative research for diffusion of innovation in the primary healthcare centers. **Journal of Modelling in Management**, v. 10, n. 1, p. 105-117, 2015.

BELADY, L. A.; LEHMAN, M. M. A model of large program development. **IBM System Journal**, v. 15, n. 3, p. 225-252, 1976.

COOK, S. et al. Evolution in software systems: foundations of the SPE classification scheme. **Journal of Software Maintenance and Evolution: Research and Practice**, v. 18, n. 1, p. 1-35, 2006.

CUNNINGHAM, W. The WyCash portfolio management system. In: ACM SIGPLAN OOPS Messenger. **Anais...** 1993.

CURTIS, B.; SAPPIDI, J.; SZYNKARSKI, A. Estimating the Principal of an Application's Technical Debt. **IEEE Software**, v. 29, n. 6, p. 34-42, 2012.

DAVIS, F. Perceived Ease of Use, and User Acceptance of Information Technology. **MIS Quarterly**, v. 13, n. 3, p. 319-340, 1989.

DAVIS, F. D.; BAGOZZI, R. P.; WARSHAW, P. R. **User Acceptance of Computer Technology:** A Comparison of Two Theoretical Models Management Science, 1989. Disponível em: <https://www.researchgate.net/profile/Richard_Bagozzi/publication/227446117_User_Acceptance_of_Computer_Technology_A_Comparison_of_Two_Theoretical_Models/links/57c85fa208ae9d640480e014/User-Acceptance-of-Computer-Technology-A-Comparison-of-Two-Theoretical-Models.pdf>. Acesso em 15 mar. 2017.

DELONE, W. H.; MCLEAN, E. R. The DeLone and McLean Model of Information Systems Success: A Ten-Year Update. **Journal of Management Information Systems**, v. 19, n. 4, p. 9-30, 2003.

FORRESTER, J. **Industrial Dynamics**. US: Productivity Press, 1961.

GEORGANTZAS, N. C.; KATSAMAKAS, E. G. Information systems research with system dynamics. **System Dynamics Review**, v. 24, n. 3, p. 247-264, 2008.

GERALDI, J. G.; MAYLOR, H.; WILLIAMS, T. Now, let's make it really complex (complicated): A systematic review of the complexities of projects. **International Journal of Operations & Production Management**, v. 31, n. 9, p. 966-990, 2010.

HERRAIZ, I. et al. The evolution of the laws of software evolution: A discussion based ona systematic literature review. **ACM Computing Surveys**, v. 46, n. 2, p. 1-28, 2013.

INCOSE. **Systems Engineering Handbook:** A Guide for System Life Cycle Processes and Activities. 4th. ed. US: Wiley, 2015.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. **ISO/IEC 14764:2006** - Software Engineering - Software Life Cycle Processes - Maintenance, 2006.

KING, S. F.; BURGESS, T. F. Beyond critical success factors: A dynamic model of enterprise system innovation. **International Journal of Information Management**, v. 26, n. 1, p. 59-69, 2006.

KRUCHTEN, P.; NORD, R. L.; OZKAYA, I. Technical Debt: From Metaphor to Theory and Practice. **IEEE Software**, v. 29, n. 6, p. 18-21, 2012.

LEHMAN, M. Programs, life cycles, and laws of software evolution. **Proceedings of the IEEE**, v. 68, n. 9, p. 1060-1076, 1980.

LEHMAN, M. Laws of software evolution revisited. EWSPT'96 Proceedings of the 5th European Workshop on Software Process Technology. **Anais...**: Lecture Notes in Computer Science. London, UK: Springer-Verlag, 1996

LEHMAN, M.; FERNÁANDEZ-RAMIL, J. C. Software Evolution. In: MADHAVJI, N. H.; RAMIL, J. F.; PERRY, D. (Eds.). **Software Evolution and Feedback**. Chichester, UK: John Wiley & Sons, Ltd, 2006. p. 7-40.

LI, Z.; AVGERIOU, P.; LIANG, P. A systematic mapping study on technical debt and its management. **Journal of Systems and Software**, v. 101, p. 193-220, 2015.

MORECROFT, J. D. A critical review of diagramming tools for conceptualizing feedback system models. **Dynamica**, v. 8, n. 1, p. 20-29, 1982.

NAN, N. Capturing Bottom-Up Information Technology Use Processes: A Complex Adaptive Systems Model. **MIS Quarterly**, v. 35, n. 2, p. 505-532, 2011.

RAMASUBBU, N.; KEMERER, C. Managing Technical Debt in Enterprise Software Packages. **IEEE Transactions on Software Engineering**, v. 5589, n. c, p. 1-1, 2014.

RAMASUBBU, N.; KEMERER, C. F.; WOODARD, C. J. Managing Technical Debt: Insights from Recent Empirical Evidence. **IEEE Software**, v. 32, n. 2, p. 22–25, 2015.

SAUER, C. **Why Information Systems Fail:** A Case Study Approach. Oxfordshire, UK: Alfred Waller, 1993.

SENGE, P. M. **The Fifth Discipline:** The Art & Practice of The Learning Organization. Revised & ed. [s.l.] Doubleday, 2006.

SOMMERVILLE, I. Large-scale complex IT systems. **Communications of the ACM**, v. 55, n. 7, p. 71, 2012.

STERMAN, J. **Business Dynamics:** Systems Thinking and Modeling for a Complex World. US: McGraw-Hill/Irwin, 2000.

STOYENKO, A. Engineering complex computer systems: a challenge for computer types everywhere. I. Let's agree on what these systems are. **Computer**, v. 28, n. 9, p. 85-86, 1995.

WEAVER, W. Science and Complexity. **American Scientist**, v. 36, p. 536, 1948.

WHITNEY, K. M.; DANIELS, C. B. The Root Cause of Failure in Complex IT Projects: Complexity Itself. **Procedia Computer Science**, v. 20, p. 325-330, 2013.

WOODSIDE, C. M. A mathematical model for the evolution of software. **Journal of Systems and Software**, v. 1, p. 337-345, 1979.

YEO, K. T. Critical failure factors in information system projects. **International Journal of Project Management**, v. 20, n. 3, p. 241-246, 2002.