Conhecendo Sistemas Legados através de Métricas de Software

Cristiane Soares Ramos
Politec Ltda
cristianer@bsb.politec.com.br

Kathia Marçal de Oliveira, Nicolas Anquetil
Universidade Católica de Brasília
{kathia, anquetil}@ucb.br

Resumo

A terceirização tem se tornado uma prática comum na indústria de software. As organizações rotineiramente subcontratam o serviço de manutenção de seus sistemas em empresas especializadas. Um grande desafio para essas empresas é rapidamente avaliar e conhecer os sistemas que elas irão manter para poder melhor definir seus contratos. Nesse contexto, este artigo apresenta um modelo de métricas para avaliar a complexidade de sistemas legados, definido com a abordagem *GQM (goal-question-metric)*. Os resultados iniciais da aplicação desse modelo em sistemas reais em Cobol são também apresentados.

Palavras-chave: avaliação de sistemas legados, métricas, manutenção de software.

Abstract

Outsourcing has become common practice in the software industry. Organizations routinely subcontract the maintenance of their software assets to specialized companies. A great challenge for these companies, is to rapidly evaluate the quality of the systems they will have to maintain so as to define their contracts. This paper presents a framework of metrics to evaluate the complexity of a legacy software system defined using the GQM (goal-question-metric) approach. Some intial results from the application of this framework in Cobol systems are also presented.

Key-words: legacy system evaluation, software metrics, software maintenance.

1. Introdução

A maioria dos estudos, na área de engenharia de software, tem se preocupado com as vantagens e desvantagens da terceirização sob o ponto de vista da empresa contratante [1]. No entanto, a aceitação de um projeto pela organização mantenedora envolve alguns riscos, que têm recebido pouca atenção [1]. A falta de conhecimento prévio sobre o sistema a ser mantido faz com que as organizações mantenedoras assumam compromissos que são difíceis de serem cumpridos, comprometendo assim a lucratividade do serviço e/ou a confiança do cliente.

Diferentes normas e modelos têm sido propostos para avaliação da qualidade de produtos (como a ISO 12219 [2] e a ISO 9126 [3]). No entanto, não existe uma proposta especificamente destinada à avaliação de sistemas legados antes de iniciar uma manutenção. A própria norma ISO9126 [3] define características e métricas para avaliar manutenção com dados coletados durante a manutenção como uma forma de monitorar a evolução do sistema. Desta forma se faz necessário definir como avaliar os sistemas legados a fim de apoiar empresas mantenedoras a melhor definir seus contratos de manutenção.

Neste artigo apresentamos um modelo de métricas, que foi definido utilizando a abordagem *Goal-Question-Metric* [4] para apoiar o melhor entendimento sobre um sistema

legado no que se refere a seu código e documentação de forma a fornecer informações para apoiar uma empresa na definição de seus contratos de manutenção.

Nas próximas seções são introduzidos brevemente conceitos e características referentes à manutenção (seção 2) e à métricas de software (seção 3). Na seção 4 será apresentado o modelo de métricas para avaliação de sistemas legados através da aplicação da abordagem GQM. A seção 5 mostra os resultados obtidos através da aplicação prática dessas métricas em sistemas em Cobol. Finalmente, na seção 6 são apresentadas as conclusões deste trabalho.

2. Manutenção de software

A manutenção é definida como a totalidade de atividades requeridas para prover suporte de custo efetivo para um sistema de software. As atividades são realizadas tanto durante o estágio de pré-entrega quanto no estágio de pós-entrega [5]. A manutenção é uma atividade inevitável e pode vir a ser a mais longa fase no ciclo de vida do sistema, isso pode ser justificado através das leis de evolução do software [5,6]. Algumas delas são:

- **Mudança contínua**: os sistemas devem ser continuamente adaptados senão eles se tornarão progressivamente insatisfatórios;
- **Aumento da complexidade**: o sistema aumenta sua complexidade conforme ele evolui, a não ser que algo seja feito para controlar tal complexidade;
- **Crescimento contínuo**: o conteúdo funcional de um sistema deve crescer continuamente para manter a satisfação do usuário durante a sua vida útil.

No contexto da avaliação de sistemas legados, são apresentados alguns fatores que dificultam e encarecem a manutenção de software [5, 7, 8]:

- Muitas vezes é excepcionalmente difícil entender o programa "de outra pessoa". Os programas são mal documentados e os algoritmos são mal escritos;
- A documentação não existe, e quando existe muitas vezes ela não é compreensível e consistente com o código fonte, neste caso ela não tem valor;
- Freqüentemente o software não é projetado e nem está preparado para sofrer mudanças.

3. Métricas de software

Segundo Fenton e Pfleeger [9] medição de software é o processo através do qual números e símbolos são atribuídos do mundo real de forma a tornar possível caracterizar cada entidade através de regras claramente definidas. A medição se dá através da aplicação de métricas de processos, produtos e serviços. Uma métrica pode ser definida, por sua vez, com uma "definição matemática, algorítmica ou função usada para obter uma avaliação quantitativa de um produto ou processo" [10]. Segundo a norma ISO 9126 [3], métrica é a composição de métodos para medição e escalas de medição.

Essas escalas de medição são formas de mapeamento para que, por meio da manipulação de dados numéricos, seja possível entender o comportamento das entidades. Existem diferentes tipos de escalas[9,11]:

- **Nominal:** que provê um nome ou um valor para um atributo, no entanto, a ordem dos valores não tem nenhum significado para a sua interpretação..

- **Ordinal:** que os resultados estão em uma determinada ordem (ascendente ou descendente), mas a distância entre os pontos dessa escala não tem significado.
- **Intervalo:** que preserva a importância da ordem dos resultados, e possui informações sobre o tamanho dos intervalos que separaram os pontos da escala.
- **Racional:** semelhante a escala de intervalo, no entanto ela representa também as proporções entre as entidades, possui o marco zero (indicador da falta do atributo).
- **Absoluta:** feita através da contagem do número de elementos de uma entidade.

Para que a medição seja efetiva, torna-se necessário focá-la nos objetivos a serem alcançados com tais medições [4]. Pfleeger [12] destaca que é necessário deixar claras as necessidades para poder melhor conhecer os objetivos das medições. O uso de modelos pode guiar a definição de aplicação de medições a serem realizadas [12], um modelo comumente utilizado é o GQM – *Goal Question Metrics* [4].

A abordagem do método GQM (*Goal-Question-Metric*) é baseada na premissa de que uma organização para medir deve primeiro especificar os seus próprios objetivos e os objetivos de seus projetos, então deve-se traçar os objetivos para os dados que os definem operacionalmente, e finalmente prover um *framework* para interpretação dos dados, respeitando os objetivos estabelecidos anteriormente [4]. O GQM considera um modelo com três níveis:

- **Conceitual** – nível no qual são definidos os objetivos de medição. O objetivo é definido para um objeto (produto, processo, ou recurso).
- **Operacional** – nível no qual são definidas questões para caracterizar um caminho para alcançar um determinado objetivo.
- **Quantitativo** – nível no qual são definidas as métricas que usam um conjunto de dados (objetivos ou subjetivos) associados com cada questão de forma quantitativa.

A fases de definição do GQM segundo [13], são:

- **Planejamento**: fase na qual é definida a equipe GQM, escolhida a área de melhoria e são preparadas e motivadas as pessoas envolvidas.
- **Definição**: fase na qual são definidos e documentados os objetivos, questões e métricas;
- **Coleta de dados**: fase na qual os dados são coletados;
- **Interpretação**: fase na qual os dados coletados são processados em relação às métricas definidas, gerando os resultados da medição, que provêem respostas para as questões definidas e posteriormente os objetivos podem ser avaliados.

4. Definição de métricas para avaliação de sistemas legados

Utilizamos a abordagem GQM para definir um conjunto de métricas que permitisse avaliar um sistema legado em um espaço curto de tempo, ou seja, em um tempo aceitável que se permita à empresa poder definir seus contratos com o cliente. Para uma empresa mantedora o custo de uma manutenção pode vir do grande número de modificações solicitadas periodicamente, da urgência das manutenções, da dificuldade intrínseca da própria atividade e da dificuldade de entender e/ou modificar o sistema. Nosso foco ao definir o

GQM foi nesse último aspecto. Ou seja, como auxiliar a empresa a conhecer o sistema que vai manter.

Para definir esse GQM seguimos o conjunto de fases apresentadas anteriormente. Na fase de *planejamento* definimos a equipe GQM que foram os autores desse trabalho e duas pessoas especialistas em manutenção (um gerente e um mantenedor). Além disso, explicitamos nosso objetivo de negócio, ou seja, permitir à empresa mantenedora avaliar a complexidade de um sistema legado de forma a apoiar na definição de seus contratos. Um importante requisito para o plano de manutenção foi dar preferência às métricas automatizáveis quando possível. A razão para isso é que uma empresa mantenedora geralmente não dispõe de muito tempo para analisar o sistema antes de definir o contrato.

Na fase de *definição* definimos dois objetivos de medição: avaliar a documentação e o código do sistema. Para cada objetivo definimos questões de avaliação e métricas para avaliá-las. A seguir são apresentados os dois objetivos de medição, assim como suas questões e métricas correspondentes.

4.1 Avaliação da documentação do sistema

O primeiro objetivo de medição definido foi para avaliar a documentação do sistema legado. A figura 1 apresenta a formalização desse objetivo segundo a estrutura proposta por [13].

Analisar a: documentação do sistema
Com o propósito de: avaliar
Com respeito a: completude e facilidade de entendimento
Do ponto de vista: analista de sistemas e do desenvolvedor
No contexto da: organização mantenedora.

Figura 1 – Objetivo de medição

Três questões foram derivadas para atender esse objetivo:

Questão 1: Qual o nível de documentação do sistema?
Questão 2: A documentação pode ser entendida pela equipe mantenedora?
Questão 3: Qual a consistência da documentação?

Para definir as métricas que respondessem essas questões tivemos que considerar o tipo de documentação que seria avaliada nesses sistemas. Definimos, então, que por nosso interesse se referir a sistemas legados, deveríamos considerar a documentação utilizada nesses sistemas que, geralmente, considerando a idade dos sistemas legados, é proveniente da análise estruturada. Consideramos então avaliar: o diagrama de contexto, diagrama de fluxo de dados nível 0, o modelo de dados (lógico/físico) e a especificação de requisitos.

Cada métrica foi definida usando um tipo de escala específico conforme definido na seção anterior. Algumas métricas foram definidas a partir da discussão entre os membros da equipe GQM e outras foram selecionadas da literatura. Para auxiliar na interpretação das medições coletadas para as métricas, propomos quando necessário uma escala ordinal que permita a melhor avaliação. As figuras 2, 3 e 4 apresentam as métricas definidas para as questões 1, 2 e 3 respectivamente.

Questão 1: Qual o nível de documentação do sistema?		
Métrica	**Fórmula**	**Interpretação**
M1.1.1) Porcentagem de elementos do diagrama de contexto que estão documentados	X = ((A1 + B1) / (A + B)) * 100 Onde: A =Total de entidades Externas B = Total de fluxos de dados A1 = Total de entidades externas documentadas B1 = Total de fluxos de dados documentados	00% a 25% - Documentação insuficiente 25% a 50% - pouco documentado 50% a 80% - moderadamente documentado. 80% a 100% - bem documentado.
M1.1.2) Porcentagem de elementos do DFD nível zero que estão documentados	X = ((A1+B1+C1+D1) / (A+B+C+D)) * 100 Onde: A = Total de entidades externas B = Total de fluxos de dados C = Total de processos D = Total de depósitos de dados A1 =Total de entidades externas documentadas B1 = Total de fluxos de dados documentados C1 = Total de processos documentados D1 = Total de depósitos de dados documentados	00% a 25% - Documentação insuficiente 25% a 50% - pouco documentado 50% a 80% - moderadamente documentado. 80% a 100% - bem documentado.
M1.1.3) Porcentagem de elementos do modelo físico de dados que estão documentados	X = ((A1 + B1 + C1) / (A + B + C)) * 100 Onde: A = Total de tabelas B = Total de colunas C = Total de domínios A1 = Total de tabelas documentadas B1 = Total de colunas documentas C1 = Total de domínios documentados	00% a 25% - Documentação insuficiente 25% a 50% - pouco documentado 50% a 80% - moderadamente documentado. 80% a 100% - bem documentado.
M1.1.4) Porcentagem de elementos do modelo entidade relacionamento que estão documentados	X = ((A1 + B1) / (A + B)) * 100 Onde: A = Total de entidades B = Total de atributos A1 = Total de entidades documentadas B1 = Total de atributos documentados	00% a 25% - Documentação insuficiente 25% a 50% - pouco documentado 50% a 80% - moderadamente documentado. 80% a 100% - bem documentado.
M1.1.5) Indicador do nível de detalhamento dos requisitos	X = A Onde: A= (3) Bem detalhado (2) Detalhado (1) Pouco detalhado (0) Insuficiente	(3) Bem detalhado (2) Detalhado (1) Pouco detalhado (0) Insuficiente
M1.1.6) Indicador de qualidade dos requisitos	X = A Onde: A = (3) Excelente qualidade (2) Boa qualidade (1) Pouca qualidade (0) Insuficiente	(3) Excelente qualidade (2) Boa qualidade (1) Pouca qualidade (0) Insuficiente
M1.1.7) Porcentagem de descrição de requisitos funcionais que possuem descrição das regras de negócio	X = A/B * 100 Onde: A = Total de requisitos funcionais registrados na especificação de requisitos que possuem descrição das regras de negócio; B = Total de requisitos funcionais registrados na especificação de requisitos.	00% a 25% - insuficiente 25% a 50% - baixo grau de descrição. 50% a 80% - descrição moderada 80% a 100% - alto grau de descrição

Figura 2 – Métricas para Objetivo 1- Questão 1

Questão 2: A documentação pode ser entendida pela equipe mantenedora?		
Métrica	**Fórmula**	**Interpretação**
M1.2.1) Indicador de facilidade de entendimento do diagrama de contexto	X = ((A + B) / (A1 + B1)) * 100 Onde: A = Total de entidades externas que possuem descrição B = Total de fluxos de dados que possuem descrição A1 = Total de entidades externas cuja descrição pode ser compreendida B1 = Total de fluxos de dados cuja descrição pode ser compreendida	00% a 25% - compreensão insuficiente 25% a 50% - baixo grau de compreensão. 50% a 80% - compreensão moderada 80% a 100% - alto grau de compreensão
M1.2.2) Indicador de facilidade de entendimento do DFD nível zero [14, "Understandability metrics" p.30]	X = ((A+B+C+D) / (A1+B1+C1+D1)) * 100 Onde: A = Total de entidades externas que possuem descrição B = Total de fluxos de dados com descrição C = Total de processos com descrição D = Total de depósitos com descrição A1 = Total de entidades externas cuja descrição pode ser compreendida B1 = Total de fluxos de dados cuja descrição pode ser compreendida C1 = Total de processos cuja descrição pode ser compreendida D1 = Total de depósitos de dados cuja descrição pode ser compreendida	00% a 25% - compreensão insuficiente 25% a 50% - baixo grau de compreensão. 50% a 80% - compreensão moderada 80% a 100% - alto grau de compreensão
M1.2.3) Indicador de facilidade de entendimento do dicionário do modelo físico de dados [14, "Understandability metrics" p.30]	X = ((A + B + C) / (A1 + B1 + C1)) * 100 Onde: A = Total de tabelas com descrição B = Total de colunas com descrição C = Total de domínios com descrição A1 = Total de tabelas cuja descrição pode ser compreendida B1 = Total de colunas cuja descrição pode ser compreendida C1 = Total de domínios cuja descrição pode ser compreendida	00% a 25% - compreensão insuficiente 25% a 50% - baixo grau de compreensão. 50% a 80% - compreensão moderada 80% a 100% - alto grau de compreensão
M1.2.4) Indicador de facilidade de entendimento do dicionário do modelo entidade relacionamento [14, "Understandability metrics" p.30]	X = ((A + B + C) / (A1 + B1 + C1)) * 100 Onde: A = Total de entidades com descrição B = Total de atributos com descrição C = Total de domínios com descrição A1 = Total de entidades cuja descrição pode ser compreendida B1 = Total de atributos cuja descrição pode ser compreendida C1 = Total de domínios cuja descrição pode ser compreendida	00% a 25% - compreensão insuficiente 25% a 50% - baixo grau de compreensão. 50% a 80% - compreensão moderada 80% a 100% - alto grau de compreensão
M1.2.5) Indicador de facilidade de entendimento da especificação de requisitos funcionais [14, "Understandability metrics" p.30]	X = (A / B) * 100 Onde: A = Total de requisitos documentados B = Total de requisitos que podem ser compreendidos	00% a 25% - compreensão insuficiente 25% a 50% - baixo grau de compreensão. 50% a 80% - compreensão moderada 80% a 100% - alto grau de compreensão

Figura 3 – Métricas para Objetivo 1- Questão 2

Questão 3: Qual a consistência da documentação?		
Métrica	**Fórmula**	**Interpretação**
M1.3.1) Porcentagem de requisitos funcionais registrados na especificação de requisitos que possuem descrição	X = (A/B) * 100 Onde: A = Total de requisitos registrados no relatório de requisitos funcionais do software, que possuem descrição. B = Total de requisitos registrados no relatório de requisitos funcionais.	00% a 25% - Descrição insuficiente 25% a 50% - baixo grau de descrição. 50% a 80% - descrição moderada 80% a 100% - alto grau de descrição
M1.3.2) Porcentagem de integrações com outros sistemas na especificação de requisitos que estão consistentes com o diagrama de contexto.	X = A/B * 100 Onde: A = Total de integrações identificadas na especificação de requisitos; B = Total de integrações identificadas no diagrama de contexto através das entidades externas.	00% a 25% - Consistência insuficiente 25% a 50% - pouca consistência. 50% a 80% - consistência moderada 80% a 100% - boa consistência
M1.3.3) Porcentagem de integrações com outros sistemas na especificação de requisitos que estão consistentes com o DFD nível zero.	X = A/B * 100 Onde: A = Total de integrações identificadas na especificação de requisitos; B = Total de integrações identificadas no DFD nível zero através das entidades externas.	00% a 25% - Consistência insuficiente 25% a 50% - pouca consistência. 50% a 80% - consistência moderada 80% a 100% - boa consistência
M1.3.4) Porcentagem de integrações com outros sistemas, documentadas na especificação de requisitos funcionais, que estão consistentes com o manual de produção.	X = A/B * 100 Onde: A = Total de integrações identificadas na especificação de requisitos funcionais e no manual de produção. B = Total de integrações identificadas no manual de produção.	00% a 25% - Consistência insuficiente 25% a 50% - pouca consistência. 50% a 80% - consistência moderada 80% a 100% - alto grau de consistência
M1.3.5) Porcentagem de elementos do modelo físico de dados que estão consistentes com o banco de dados de produção.	X = ((A + B) / (A1 + B1)) * 100 Onde: A = Total de tabelas definidas no modelo físico de dados e no banco de dados de produção B = Total de colunas definidas no modelo físico de dados e no banco de dados de produção A1 = Total de tabelas existentes no banco de dados de produção B1 = Total de colunas existentes no banco de dados de produção	00% a 25% - Consistência insuficiente 25% a 50% - pouca consistência. 50% a 80% - consistência moderada 80% a 100% - alto grau de consistência
M1.3.6) Porcentagem de todas as entidades externas representadas no DFD nível Zero ou no diagrama de contexto que estão no DFD nível zero e no diagrama de contexto	X = (A / B) * 100 Onde: A = Total de entidades externas representadas no diagrama de contexto ou no DFD nível Zero. B = Total de entidades externas representadas no DFD nível Zero e no diagrama de contexto	00% a 25% - Consistência insuficiente 25% a 50% - pouca consistência. 50% a 80% - consistência moderada 80% a 100% - alto grau de consistência.
M1.3.7) Nível de consistência do manual de produção com o diagrama de contexto	X = A Onde: A = (3) Totalmente consistente (2) Consistente (1) Pouca consistência (0) Nenhuma consistência	(3) Totalmente consistente (2) Consistente (1) Pouca consistência (0) Nenhuma consistência
M1.3.8) Nível de consistência do manual de produção com o DFD-0	X =A Onde: A = (3) Totalmente consistente (2) Consistente (1) Pouca consistência (0) Nenhuma consistência	(3) Totalmente consistente (2) Consistente (1) Pouca consistência (0) Nenhuma consistência

Figura 4 – Métricas para Objetivo 1- Questão 3

4.2 Avaliação do código do sistema

O segundo objetivo de medição é avaliar o código fonte do sistema legado. A figura 5 apresenta a formalização desse objetivo de acordo com a estrutura proposta por [13].

Analisar o: sistema implementado
Com o propósito de: avaliar
Com respeito ao: complexidade
Do ponto de vista do: analista de sistemas e do desenvolvedor
No contexto: organização mantenedora.

Figura 5 – Objetivo de medição 2

Cinco questões foram derivadas para atender esse objetivo:

Questão 1: Qual o tamanho do sistema?
Questão 2: Qual o nível de interação no sistema?
Questão 3: Qual a complexidade do sistema implementado?
Questão 4: Qual a complexidade de interface com o usuário?
Questão 5: Qual a complexidade de interface com outros sistemas?

Para definir métricas para essas questões consideramos uma ampla revisão da literatura e o nosso entendimento do que significa um sistema ser complexo. Nesse sentido a complexidade de manutenção está relacionada à dificuldade de entender/manter um sistema considerando o número de linhas de código (total e por programa/módulo), número de tabelas que utiliza (do próprio ou de outros sistemas lidas e atualizadas), número de telas a serem manipuladas (total e por programa), quantidade de tecnologias linguagens/plataformas) envolvidas, grau de acoplamento entre os programas e a própria complexidade do código. Para definir essas métricas utilizamos novamente a experiência da equipe GQM e a literatura relevante. Para as métricas definidas para esse objetivo não foi possível definir limites de valores que nos ajudasse na interpretação. Esses valores devem ser definidos a medida em que os sistemas sejam avaliados considerando uma base histórica do mesmo. No entanto, para auxiliar na interpretação, destacamos algumas observações da literatura sobre as métricas coletadas.

As figuras 6, 7, 8, 9 e 10 apresentam as métricas das questões desse objetivo.

Questão 1: Qual o tamanho do sistema?		
Métrica	**Fórmula**	**Observações para Interpretação**
M2.1.1) Tamanho do sistema em LOC [5, 6, 8, 15, 16, 17, 18, 19]	X = A Onde: A = Total de linhas de código do sistema * não contar linhas em branco.	A manuntenabilidade tende a diminuir conforme a quantidade de linhas de código aumenta. Quanto maior o sistema, maior será o esforço requerido para realizar a manutenção. [18,19] (ver também: www.asq.org)
M2.1.2) Média de LOC por programa [20]	X = M2.1.1 / M2.1.8 Onde: M2.1.1 = Tamanho do sistema em LOC M2.1.8 = Número de programas do sistema	A manuntenabilidade tende a diminuir conforme a quantidade de linhas de código aumenta. Quanto maior o sistema, maior será o esforço requerido para realizar a manutenção. [18] (ver também: www.asq.org)
M2.1.3) Distribuição de LOC por programa.	Para cada programa, X = A Onde: A = Quantidade de linhas de código do programa.	

Figura 6 – Métricas para Objetivo 2- Questão 1

Questão 1: Qual o tamanho do sistema?		
M2.1.4) Número de tabelas próprias do sistema	X = A Onde: A = Total de tabelas próprias do sistema que são acessadas para leitura ou atualização.	
M2.1.5) Número de tabelas de outro sistema	X = A Onde: A = Total de tabelas de outros sistemas legados que são acessadas para leitura ou atualização.	Quanto maior a quantidade de tabelas acessadas, maior será a complexidade. Baseado na característica geral da APF "Modificação facilitada"
M2.1.6) Número de LOC por módulo	X = M2.1.1 / B Onde: B = Total de módulos do sistema. M2.1.1 = Tamanho do sistema em LOC	A manuntenabilidade tende a diminuir conforme a quantidade de linhas de código aumenta. Quanto maior o sistema, maior será o esforço requerido para realizar a manutenção.
M2.1.7) Distribuição de LOC por Módulo.	Para cada módulo, X = A Onde: A = Quantidade de linhas de código de todos os programas que compõe o módulo.	
M2.1.8) Número de programas do sistema	X = A Onde: A = Total de programas que compõe o sistema.	
M2.1.9) Média de programa por módulo	X = M2.1.8 / A Onde: A = Número de módulos do sistema. M2.1.8 = Número de programas do sistema	
M2.1.10) Número total de telas	X = A Onde: A = Número total de telas do sistema.	
M2.1.11) Distribuição de programas por tecnologia	Para cada tecnologia envolvida, X = M2.1.8 Onde: M2.1.8 = Número de programas do sistema	Quanto maior a quantidade de tecnologias envolvidas, mais esforço será requerido para realizar a manutenção.

Figura 6 (cont.) – Métricas para Objetivo 2- Questão 1

Questão 2: Qual o nível de interação no sistema?		
Métrica	**Fórmula**	**Observações para Interpretação**
M2.2.1) Média de tabelas próprias lidas por programa	X = A / M2.1.8 Onde: A = Número de tabelas próprias do sistema acessadas para leitura M2.1.8 = Número de programas do sistema	Quanto maior a quantidade de tabelas acessadas, maior será a complexidade. Baseado na característica geral da APF "Modificação facilitada"
M2.2.2) Média de tabelas próprias modificadas por programa	X = A / M2.1.8 Onde: A = Número de tabelas próprias do sistema acessadas para inserção, alteração ou exclusão de dados. M2.1.8 = Número de programas do sistema	Quanto maior a quantidade de tabelas acessadas, maior será a complexidade. Baseado na característica geral da APF "Modificação facilitada"
M2.2.3) Fan-out médio por programa [21]	X = A / M2.1.8 Onde: A = Total de fan-out M2.1.8 = Número de programas do sistema	Quanto maior o valor do fan-out, menor será a manutenabilidade do sistema.
M2.2.4) Distribuição fan-out por programa.	Para cada programa, X = A Onde: A = Total de fan-out do programa.	

Figura 7 – Métricas para Objetivo 2- Questão 2

Questão 3: Qual a complexidade do sistema implementado?		
Métrica	**Fórmula**	**Observações para Interpretação**
M2.3.1) Distribuição da complexidade ciclomática por programa [15, 16, 18, 21]	Para cada programa, X = A Onde: A = complexidade ciclomática do programa	A manutenabilidade diminui conforme a complexidade ciclomática aumenta [18]
M2.3.2) Média da Complexidade ciclomática por programa.	X = A / M2.1.8 Onde: A = Soma da complexidade ciclomática de todos os programas M2.1.8 = Número de programas do sistema	De 1 a 10 = simples De 11 a 20 = pouco complexo De 21 a 50 = complexo > 50 = intestável (ver: mdp.ivv.nasa.gov)
M2.3.3) Distribuição de desvios por programa [18, 21]	Para cada programa, X = A Onde: A = Total de GOTO do programa	A manutenabilidade tende a diminuir conforme a quantidade de desvios aumenta. [18]
M2.3.4) Média de desvios por programa.	X = A / M2.1.8 Onde: A = Total de GOTO de todos os programas M2.1.8 = Número de programas do sistema	A manutenabilidade tende a diminuir conforme a quantidade de desvios aumenta. [18]
M2.3.5) porcentagem de L0C comentadas do sistema [5, 8, 15, 16, 17, 18]	X = A/B Onde: A = Total de linhas comentadas do sistema B = Total de linhas do sistema *não serão consideradas linhas vazias	Quanto maior a quantidade de linhas de comentário do sistema, maior será a sua manutenabilidade. [5]
M2.3.6) Distribuição de linhas de comentário por programa	Para cada programa, X = A Onde: A = Total de linhas comentadas do programa	
M2.3.7) Número médio de biblioteca acessada por programa	X = A / M2.1.8 Onde: A = Total de bibliotecas acessadas pelo sistema M2.1.8 = Número de programas do sistema	
M2.3.8) Média de Operadores distintos do sistema [16]	X = (A + B) / M2.1.8 Onde: A = número e parâmetros utilizados em todos os programas B = número de constantes utilizadas em todos os programas M2.1.8 = Número de programas do sistema	
M2.3.9) Média de IF's aninhados do programa [16]	X = A / M2.1.8 Onde: A = Nº de If's aninhados M2.1.8 = Número de programas do sistema	
M2.3.10) Número de linguagens de programação	X = A Onde: A = quantidade de linguagens de programação utilizadas para implementar o sistema.	
M2.3.11) Distribuição de programas por linguagem de programação	Para cada linguagem de programação, X = A Onde: A = Quantidade de programas implementados em determinada linguagem de programação	
Reutilização de **M2.2.3) Fan-out médio por programa**		
M2.3.12) Complexidade intermodular e intramodular [18]	X = S + L Onde: S = complexidade intermodular = soma (fan-out^2) / n L = complexidade intramodular = número de var. entrada/saída do módulo n = número de módulos do sistema * considerar variáveis globais e parâmetros ** variáveis globais não referenciadas não devem ser consideradas	Quanto maior o valor encontrado, maior será a complexidade do sistema e menor a manutenabilidade. [16]

Figura 8 – Métricas para Objetivo 2- Questão 3

Questão 3: Qual a complexidade do sistema implementado?		
M2.3.13) Distribuição da métrica de esforço de Halstead por programa [15, 18] (ver também: mdp.ivv.nasa.gov, www.asq.org)	Para cada programa, X =V * D Onde: V = volume = (N1/2) * (N2/n2) D = dificuldade = N * Log2n n1 = número de operadores distintos n2 = número de operandos distintos N1 = número total de ocorrência de operadores N2 = número total de ocorrência de operandos N = N1 + N2 n = n1 + n2	A manutenabilidade tende a diminuir conforme o esforço aumenta [18]
M2.3.14) Distribuição do Índice de Manutenabilidade por programa [15]	X = 171 – 5.44 * ln(avVol) – 0.23 * aveV(g') – 1.62 * ln(avLOC) + 50 * sin(sqrt(2.46 * aveCM)) Onde: aveVol = Halstead Volume médio por módulo aveV(g') = complexidade ciclomática média por módulo aveLOC = média de linhas de código por módulo aveCM = média de linhas de comentários por módulo	X < 65 → baixa manutenabilidade X de 65 a 85 → manutenabilidade média X > 85 → alta manutenabilidade [15]

Figura 8 (cont.) – Métricas para Objetivo 2- Questão 3

Questão 4: Qual a complexidade de interface com o usuário?		
Métrica	**Fórmula**	**Observações para Interpretação**
Reutilização de **M2.1.10) Número total de telas**		
M2.4.1) Número médio de tela por programa	X = M2.1.10 / M2.1.8 Onde: M2.1.8 = Número de programas do sistema M2.1.10 = Número total de telas	
M2.4.2) Número médio de campo de tela por programa	X = A / M2.1.8 Onde: A = Soma de campos de entrada / saída de dados de todas as telas do sistema M2.1.8 = Número de programas do sistema	
M2.4.3) Distribuição do número de campos por tela	Para cada programa, X = A Onde: A = Soma de campos de entrada / saída de dados da tela do programa.	

Figura 9 – Métricas para Objetivo 2- Questão 4

Questão 5: Qual a complexidade de interface com outros sistemas?		
Métrica	**Fórmula**	**Observações para Interpretação**
Reutilização de **M2.1.5) Número de tabelas de outro sistema**		
M2.5.1) Número de tabelas lidas de outro sistema	X = A Onde: A = Total de tabelas de outros sistemas que são acessadas para leitura (considerar todo o sistema)	Quanto maior a quantidade de tabelas acessadas, maior será a complexidade. Baseado na característica geral da APF "Modificação facilitada"
M2.5.2) Número de tabelas modificadas de outro sistema	X = A Onde: A = Total de tabelas de outros sistemas que são acessadas para inclusão, alteração ou exclusão de dados (considerar todo o sistema)	Quanto maior a quantidade de tabelas acessadas, maior será a complexidade. Baseado na característica geral da APF "Modificação facilitada"
M2.5.3) Distribuição de tabelas acessadas de outros sistemas, por programa.	Para cada programa, X = A. Onde: A = Total de tabelas de outros sistemas que são acessadas para inclusão, alteração ou exclusão de dados.	
Reutilização de **M2.3.8) Média de Operadores distintos do sistema**		

Figura 10 – Métricas para Objetivo 2- Questão 5

5. Aplicação Prática Inicial

Este trabalho está sendo validado em uma grande empresa de software. Esta empresa possui mais de 5.500 funcionários distribuídos em cinco filiais no Brasil, sendo que duas delas possuem uma área especializada em manutenção de software.

Ainda não se têm dados para todas as métricas apresentadas acima, estamos no processo de implementação de uma ferramenta que calcule as métricas de código para Cobol, mas já foi implementado um protótipo da ferramenta que calcula alguma dessas métricas e esse protótipo foi testado sobre módulos Cobol de 5 sistemas. Algumas das métricas de documentação que são avaliadas manualmente foram também avaliadas para esses mesmos sistemas. As tabelas 1 e 2 apresentam os resultados dessa primeira aplicação.

Tabela 1 – Resultado da aplicação das métricas sobre a documentação dos sistemas

	Sist.A	**Sist.B**	**Sist.C**	**Sist.D**	**Sist.E**
Q1) Qual o nível de documentação do sistema?					
M1.1.1	58%	50%	50%	-	64%
M1.1.2	-	48%	-	-	90%
M1.1.3	2%	-	-	-	-
M1.1.5	Detalhado	Bem detalhado	Detalhado	Bem detalhado	Detalhado
M1.1.7	86%	100%	33%	47%	100%
Q3) Qual a consistência da documentação?					
M1.3.1	100%	100%	100%	100%	100%
M1.3.2	75%	-	-	-	-
M1.3.4	100%	85%	0%	-	-
M1.3.5	-	-	-	-	-
M1.3.6	-	29%	-	-	33%
M1.3.7	Consistente	Pouca consistência	Pouca consistência	-	-
M1.3.8	Consistente	Pouca consistência	-	-	-

(*) *Resultados em branco indicam que a documentação não estava disponível para consulta.*

Conforme definido pela abordagem GQM, através dos dados obtidos para as métricas, constatou-se um baixo nível de documentação dos sistemas (Questão 1, Tabela 1). Isso não é uma surpresa, a falta de documentação de sistemas legados é lendária. O único item que apresenta um quadro melhor são os requisitos (M1.1.5 e M1.1.7). Além da falta de documentação, podemos também perceber um problema de consistência dessa documentação. Com exceção do Sistema A, os quatro outros sistemas apresentam grandes falhas nesse quesito o que invalida provavelmente o uso eficaz da documentação para a manutenção e em conseqüência deve dificultar a mesma.

Dos resultados da questão 1, Tabela 2, podemos concluir que não se trata de grande sistemas (sistemas Cobol chegam rotineiramente ao milhão de LOC). Mas deve ser lembrado que foi considerado aqui apenas os módulos Cobol desses sistemas. Os programas avaliados, em média, são pequenos nos cinco sistemas (M2.1.2). O nível de interação difere em função dos sistemas, os dois primeiros têm uma pequena interação com outros sistemas ao nível dos dados (M2.1.5), enquanto o terceiro sistema tem uma maior interação de fluxo (M2.2.3). Do ponto de vista da complexidade dos sistemas, a

complexidade ciclomática (M2.3.2) varia de pouco complexa (Sistema A) a instável (sistema E) com três sistemas pouco complexos (sistemas A, C e D). Verificou-se a existência de poucos GOTOs (M2.3.4) o que indica uma boa prática de programação. A média de linhas comentadas é próxima para o 5 sistemas e se situa entre 20% (1 linha comentada em 4) e 30% (1 linha comentada em 3) o que parece uma proporção razoável. Deve ser notado, no entanto, que a qualidade e pertinência dos comentários não foi avaliada e que se trata apenas de uma medida quantitativa. A métrica de esforço de Halstead (M2.3.13) resulta em números muito altos (Sistemas B e E) que podem ser indicadores de problemas potenciais. Finalmente, a interface com outros sistemas legados é mínima nos cinco sistemas avaliados.

Tabela 2 – Resultado da aplicação das métricas sobre o código dos sistemas

	Sist.A	**Sist.B**	**Sist.C**	**Sist.D**	**Sist.E**
Q1) Qual o tamanho do sistema?					
M2.1.2	486,6	589,5	676,2	469,6	816,8
M2.1.4	19	44	8	4	24
M2.1.5	2	5	0	0	0
M2.1.6	14 110	43 623	33 135	3 757	16 336
M2.1.8 / M2.1.9 / M2.1.11	29	74	49	8	20
M2.1.10	0	0	0	0	0
Q2) Qual o nível de interação no sistema?					
M2.2.1	16	41	0	3	0
M2.2.2	18	34	8	1	25
M2.2.3	1,3	1,1	9,4	0,5	2,3
Q3) Qual a complexidade do sistema implementado?					
M2.3.2	11,7	26,4	16,4	14,0	50,6
M2.3.4	0,4	1,1	0,0	1,7	0,1
M2.3.5	29,8%	26,0%	28,8%	28,0%	20,3%
M2.3.13	32,7	1121,3	95,8	321,7	1407,4
Q5) Qual a complexidade de interface com outros sistemas?					
M2.5.1	1	4	0	0	0
M2.5.2	1	1	0	0	0

6. Conclusão

Neste artigo foi apresentada uma proposta de modelo de métricas para avaliação de sistemas legados antes de iniciar o processo de manutenção do sistema. Esta proposta se baseou no processo de definição de plano de qualidade GQM (*Goal-Question-Metric*) em que se define primeiro o objetivo do plano de qualidade, objetivo que em seguida é decomposto em questões a serem respondidas para avaliar se ele foi atendido, e finalmente, as questões são avaliadas a través de métricas.

Já foi realizada uma avaliação preliminar onde algumas das métricas propostas, tanto para a documentação, quanto para o seu código fonte foram avaliadas para cinco sistemas. Essa primeira avaliação permite tirar algumas conclusões sobre a qualidade e a provável facilidade de manutenção dos sistemas mensurados.

Essa proposta está ainda em validação e será necessária a aplicação das métricas de documentação e de código em uma base maior de sistemas para verificar se as mesmas demonstram serem viáveis no que diz respeito à avaliação de sistemas legados.

Agradecimentos

Os autores agradecem ao Centro de Sustentação de Sistemas da empresa Politec por todo apoio na realização prática deste trabalho.

Esse trabalho faz parte do Projeto Hércules, uma parceria entre a Politec e a Universidade Católica de Brasília.

Bibliografia

[1] POLO, M., PIATTINI, M., RUIZ, F., **"Using code metrics to predict maintenance of legacy programs: a case study"**, IEEE Transaction on Software Engineering, 1998.

[2] ISO 12219 "**Tecnologia de informação – Pacote de software –** Teste e requisitos de qualidade", ABNT, 1998.

[3] ISO 9126 **"Software engineering – Product quality –** Part 1", 2001

[4] BASILI, V. e ROMBACH, H. "**Goal question metric paradigm"**, Encyclopedia of software engineering – 2, 1994.

[5] PIGOSKI, T.M., "**Practical Software Maintenance**" John Wiley & Sons, Inc., 1996

[6] LEHMAN, M.M., PERRY, D.E, RAMIL, J.F, "**Implications of evolution metrics on software maintenance"**, IEEE Transaction on Software Engineering, 1998.

[7] PRESSMAN, R., **"Engenharia de software"**, Makron Books, 1995.

[8] GARCÍA, M. e ALVAREZ, J., "**Manutenability as a key factor in maintenance productivity: a case study**", IEEE Transaction on Software Engineering, 1996.

[9] FENTON, N.E, PFLEEGER, L., **"Software metrics a rigorous and practical approach",** PWS Publishing Company, 1997

[10] FENTON, N.E.; "**Software Metrics - A Rigorous Approach"**; Chapman & Hall; 1991

[11] PARK R.E., GOETHERT W.B. e FLORAC W.A. "**Goal Driven Software Measurement – a Guidebook"**, CMU/SEI-96-BH-002, Software Engineering Institute, Carnegie Mellon University, August 1996

[12] PFLEEGER, S., "**Use realistics, effective software measurement**" cap. 8, in: **"Constructing Superior Software",** Eds. CLEMENTS, Paul C., BASS Len, BELADY Les, *et al*, Software Quality Institute, 2000.

[13] SOLIGEN, R. e BERGHOUT, E., "**The goal/question/metric method** – A pratical guide for quality improvement of software development". Great Britain: Cambridge, McGraw-Hill, 1999

[14] ISO 9126 **"Software engineering – Product quality –** Part 2: external metrics**",** 2001

[15] PEARSE, T. e OMAN, P. "**Maintainability measurement on industrial source code maintenance activities",** 295-303. *Proceedings. of the International Conference on Software Maintenance*. Opio, France, October 17-20, 1995. IEEE Computer Society Press, 1995.

[16] FRENCH, V.A., "**Establishing software metric thresholds",** Nineth International Workshop on Software Measurement (IWSM'99).

[17] OLIVEIRA, E.A., GIMENES, I.S., **"Definição de métricas de produtos de *software* para uma ferramenta de workflow"**, Universidade Estadual de Maringá, Paraná, 1994.

[18] FRAPIER, M., MATWIN, S., MILI, A. "**Software metrics for predicting maintainability** - Software metrics study: Technical Memorandum 2**",** University of Ottawa, 1994

[19] POLO, M., PIATTINI, M., RUIZ, F., "Using code metrics to predict maintenance of legacy programs: a case study", IEEE International Conference on Software Maintenance (ICSM'01), pp. 202-208, Nov. 07-09, 2001, Florence, Italy.

[20] DEMEYER, S., DUCASSE, S., LANZA, M., "**A hybrid reverse engineering approach combining metrics and program visualization",** WCRE 1999 Proceedings of the Working Conference on Reverse Engineering, pp. 175-186, IEEE Computer Society Press, 1999

[21] KHOSHGOFTAAR, T.M., ALLEN, E.B., HALSTEAD, R., e TRIO G.P., "**Detection of fault-prone software modules during a spiral life cycle**", International Conference on Software Maintenance (ICSM '96), pp. 69-76, Nov. 04-08, 1996, Monterey, CA, 1996.