UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CÂMPUS CORNÉLIO PROCÓPIO
DEPARTAMENTO DE COMPUTAÇÃO
ENGENHARIA DE COMPUTAÇÃO

ALLAN VICTOR MORI

DESENVOLVIMENTO DE UM PLUG-IN PARA A CONTAGEM DE PONTOS POR FUNÇÃO NO SOFTWARE ASTAH

TRABALHO DE CONCLUSÃO DE CURSO

CORNÉLIO PROCÓPIO
2015

ALLAN VICTOR MORI

DESENVOLVIMENTO DE UM PLUG-IN PARA A CONTAGEM DE PONTOS POR FUNÇÃO NO SOFTWARE ASTAH

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina Trabalho de Conclusão de Curso 2, do curso de Engenharia de Computação da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito para a obtenção do título de Bacharel em Engenharia de Computação.

Orientador: Prof. Dr. José Augusto Fabri

CORNÉLIO PROCÓPIO

2015

AGRADECIMENTOS

Agradeço primeiramente a Deus por iluminar meus passos desde sempre, e depois a meus pais, Dorival e Mariluci, meu irmão Alaor e minha namorada Fabiana, os quais sempre ofereceram amor e compreensão ao longo das minhas jornadas e suporte principalmente nos meus últimos 5 anos permitindo que eu concretizasse um dos meus maiores sonhos.

Também agradeço ao meu Orientador José Augusto Fabri, por ter confiado e aberto portas em projetos científicos que enriqueceram minha vivência acadêmica e pessoal.

O meu muito obrigado a todos os mestres e servidores da Universidade Tecnológica Federal do Paraná - Campus Cornélio Procópio, que me ensinaram fundamentos que levarei sempre comigo.

Um obrigado especial para todos aqueles que me marcaram e dividiram conhecimento, tempo, diversão, comida, abrigo, remédios, caronas e treinos, meus grandes colegas de cursos e salas.

Muito Obrigado à todos!

Que Deus os abençoe grandemente!

RESUMO

MORI, Allan Victor. **Desenvolvimento de um Plug-in para a Contagem de Pontos por Função no Software Astah**. 2015. 70 f. Trabalho de Conclusão de Curso (Graduação) – Engenharia de Computação. Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2015.

A realização de um planejamento eficaz de um software é decisivo para seu desenvolvimento, é necessário estimar, medir e controlar: o tempo, o custo e a complexidade. É preciso ir além da idealização em diagramas para a execução, seguir padrões de qualidade e desenvolvimento embasados em arquiteturas de software, que fazem com que o projeto tenha suas funcionalidades especificadas cumpridas. Outro ponto importante para o sucesso de projetos de softwares é cumprir com o prazo de entrega, neste contexto é considerado a utilização de técnicas que permitam estimar o tamanho do software, neste trabalho é apresentado uma ferramenta que automatiza a contabilização dos Pontos por Função de um projeto criado no programa de modelagem UML, o Astah Professional, que é utilizado para arquitetar sistemas de software. O Astah permite o desenvolvimento de plug-ins baseados em Java, fazendo-se uso do ambiente de desenvolvimento Eclipse.

Palavras-chave: Pontos por Função no Astah. Métricas de Software. Plug-in. Pontos por Função. Astah.

ABSTRACT

MORI, Allan Victor. **Development of a Plug-in for Count the Function Points in Astah Software**. 2015. 70 f. Trabalho de Conclusão de Curso (Graduação) – Engenharia de Computação. Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2015.

Conducting an effective planning software is critical to their development, it is necessary to estimate, measure and control: time, cost and complexity. It needs to go beyond the idealization in diagrams for execution, following standards of quality and grounded developing software architectures, which make the project has met its specified functionality. Another important point to the success of a softwares projects, is to comply with the deadline, in this context is considered the use of techniques to estimate software size. This work presents a tool that automates the recording of Function Points in a project created in the UML modeling program, Astah Professional, which is used to engineer software systems. The Astah allows the development of Java-based plug-ins, making use of the Eclipse development environment.

Keywords: Function Points on Astah, Software Metrics, Plugin, Function Points, Astah.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE SIGLAS

ALI	Arquivo Lógico Interno
AIE	Arquivo de Interface Externo
API	Application Programming Interface (Interface de Programação de Aplicativos)
CE	Consulta Externa
EE	Entrada Externa
ID	Identificador
DER	Diagrama Entidade-Relacionamento
UML	Unified Modeling Language (Linguagem de Modelagem Unificada)
RLR	Registro Lógico Referenciado
SDK	Software Development Kit (Conjunto de Desenvolvimento de Programas)
SE	Saída Externa
XML	Extensible Markup Language (Linguagem de Marcação Extensiva)

LISTA DE ACRÔNIMOS

IFPUG	Grupo Internacional de Usuários de Pontos por Função
UTFPR	Universidade Tecnológica Federal do Paraná

SUMÁRIO

1 INTRODUÇÃO

No início da computação o hardware era utilizado para propósito geral, enquanto o software era projetado sobre medida e tinha uma distribuição relativamente limitada, porém houve a evolução do hardware, de processadores à válvula até os dispositivos microeletrônicos, aumentando significativamente o número de operações que um computador poderia realizar.

Desta forma, tornou-se necessário mais softwares para funções diferentes, levando-o a se tornar produto industrial. Para suprir a demanda, o software passou a ser desenvolvido em larga escala, evidenciando problemas passados e novos, como a ausência de documentação, prazos imprecisos, ou ainda atrasos e programas que não atendiam as necessidades de seus contratantes.

Na década de 80, havia uma grande desconfiança sobre os programas desenvolvidos, que levavam grandes corporações a tomar decisões de não automatizar seus empreendimentos, pois o desenvolvimento era realizado de forma desordenada ou sem um processo bem definido, ou ainda sem medidas de controle e observação dos estágios de produção (PRESSMAN, 1992).

Buscando solucionar os problemas e trazer estabilidade para o desenvolvimento de grandes softwares, surge a Engenharia de Software, que agrega os conceitos da Engenharia à produção de Software. Esta área estuda os passos envolvidos no desenvolvimento de software, sendo fundamentada sobre os conceitos de processos e métricas de programas.

Através da construção de artefatos como diagramas e imagens representa objetivamente os requisitos de seus projetos, para assim manter um caminho bem estruturado até a finalização do software pretendido (PAULA FILHO, 2009).

Sobre este contexto os autores Pressman e Paula Filho explicam sobre a necessidade do uso de diagramas, mostrando que são essenciais para a representação das ideias e conceitos na Engenharia de Software, os diagramas são construídos utilizando a linguagem UML (*Unified Modeling Language*) que possui regras para facilitar a confecção e padroniza-los. Neste contexto, existem ferramentas que auxiliam na criação de projetos de software, dentre elas:

a) StarUML (STARUML, 2015), segundo Macoratti tem por objetivo oferecer uma ferramenta de modelagem e também uma plataforma que seja um substituto convincente para as ferramentas comerciais (MACORATTI, 2015).

b) ArgoUML (ARGOUML, 2015), É uma ferramenta que é executada em plataforma Java (JAVA, 2015) e também *Open-Source*.

c) Astah (ASTAH, 2015), ferramenta utilizada no meio acadêmico e profissional, que se diferencia das outras pelo número de recurso e as possibilidades de desenvolvimento de novas funcionalidades.

Ainda sobre o Astah, este oferece os recursos necessários para a modelagem de projetos de softwares completos, segundo a página *online* do aplicativo este é, “uma ferramenta muito utilizada para arquitetar sistemas de forma intuitiva e inovadora, possuindo versões diferenciadas para fins específicos” (ASTAH, 2015).

A versão mais completa, a *Professional*, se destaca ao oferecer a possibilidade de criar mapas mentais e Diagramas Entidade-Relacionamento. Pelo fato da empresa que a mantém, Change Vision, oferecer um pacote de desenvolvimento da ferramenta é possível que a comunidade colabore com a criação de novas funcionalidades, adicionando plug-ins ao Astah.

1.1 MOTIVAÇÃO

Segundo Paula Filho (2009, p. 5) alguns dos problemas encontrados em softwares mal planejados são: softwares caros, baixa qualidade, e entrega fora do prazo. Sobre este cenário, o relatório Chaos de 2013 mostra que os principais problemas que podem ser encontrados na prática em projetos de software são: projetos que ultrapassaram a data de entrega e projetos que não estavam completos. Esta perspectiva permite visualizar um mal planejamento para o desenvolvimento eficiente desses projetos, evidenciando que as estimativas de prazo e de custo são frequentemente imprecisas, ou ainda mostrando que a

produtividade na área de software não tem acompanhado a demanda por seus serviços, tendo como resultado uma baixa qualidade de software (CHAOS REPORT, 2013).

No relatório mencionado, publicado pelo Standish Group, apenas 39% (2013, p. 5) dos projetos são completados tendo os prazos respeitados e os requisitos implementados. Este fato mostra que os projetos não atendem as premissas de qualidade como, implementação dos requisitos explícitos e implícitos e entrega dentro do prazo, evidenciando que os conceitos da Engenharia de Software não são sempre aplicados na prática.

Com o objetivo de sistematizar a aplicação dos conceitos de Pontos por Função, automatizar sua contabilização, aproximar a teoria de métricas de software à prática, uma nova funcionalidade foi desenvolvida para auxiliar os desenvolvedores de software. Essa funcionalidade auxilia aos desenvolvedores escolhas de direções pautadas em análise de dados reais do projeto. Os Pontos por Função utiliza o conceito de que cada função possui uma complexidade. No desenvolvimento dessa nova funcionalidade para o Astah, foi utilizado uma abordagem orientada à objeto, permitindo este trabalho ser expandido, possibilitando também a outros desenvolvedores interessados em construir suas próprias ferramentas.

1.2 OBJETIVO

Este trabalho visa o desenvolvimento de um plug-in para automatizar a contagem de pontos por função no software Astah.

1.3 ORGANIZAÇÃO DO TEXTO

Este trabalho foi divido em 5 capítulos, no Capítulo 1 é apresentado o contexto sobre a Engenharia de Software, área em que encontra-se o tema e quais as motivações deste trabalho, sendo acompanhado pelo objetivo.

No Capítulo 2 são apresentados os conceitos relacionados, as tecnologias utilizadas e os fundamentos aos quais o trabalho se refere ao longo da solução implementada.

No Capítulo 3, é mostrado as arquiteturas seguidas para obter o Plugin Function Points 1.0 até atingir as funcionalidades especificadas nos fundamentos apresentados.

Os resultados obtidos com a ferramenta desenvolvida neste trabalho podem ser visualizados no Capítulo 4 que acompanha um exemplos de utilização do plug-in.

No Capítulo 5, considerações são ponderadas sobre possíveis melhorias do plug-in e trabalhos que poderão ser desenvolvidos utilizando os fundamentos abordados.

2 FUNDAMENTAÇÃO TEÓRICA

Este trabalho utiliza o Diagrama de Casos de Uso, o Diagrama Entidade-Relacionamento que estão inseridos na UML, além desses também é apresentado os conceitos iniciais sobre uma técnica de métrica de software, os Pontos por Função. Neste capítulo serão abordado os tópicos mencionados acima e alguns dos trabalhos relacionados à utilização dos conceitos.

2.1 REVISÃO BIBLIOGRÁFICA

Na Engenharia de Software existe uma constante busca por qualidade e evolução dos projetos atendendo aos requisitos desejados pelo cliente. Devido a essa premissa faz-se necessário a utilização de métricas e formas de manter os processos nas especificações de requisitos e dentro do prazo combinado. Guiando o projeto pelas etapas de desenvolvimento na realidade da empresa que o desenvolve. O relatório Chaos 2013 promove ideias e técnicas para obtenção de resultados em projetos de software, os pontos chave desta edição do relatório explicitam o gerenciamento e também a aproximação do cliente para a melhoria das estatísticas de sucesso. Isso evidencia que ferramentas de controle e comunicação contribuem na prática de desenvolvimento de software para o sucesso dos projetos (CHAOS, 2013).

A utilização de métricas no processo de desenvolvimento de software acrescenta uma variável de controle, que permite visualizar, o tamanho do projeto que será desenvolvido, o avanço do projeto, ou qual a complexidade de uma manutenção será realizada. No trabalho desenvolvido por Macedo é apresentado uma ferramenta de contabilização dos pontos por função para aplicativos móveis. Essa ferramenta tem foco em técnicas que estimam sobre interfaces gráficas, abrindo caminhos para utilização da técnica em smartphones, beneficiando o planejamento voltado a este tipo de desenvolvimento de software (MACEDO, 2003).

Na área de gerenciamento de projetos de software, atrelado à métricas, o trabalho de Bomfim Junior (2011, p. 6) contribui na criação de um banco de projetos,

que podem ser utilizados como base de dados históricos e tamanhos de projeto, com a finalidade na comparação, para obter estimativas mais refinadas e realistas (BOMFIM JUNIOR, 2011).

Os trabalhos citados relacionam o gerenciamento no desenvolvimento de softwares e as métricas como pontos principais para o desenvolvimento eficiente e planejados de projetos de software, mostrando uma perspectiva do que está sendo realizado utilizando os conceitos e as ferramentas da Engenharia de Software. Alguns dos conceitos utilizados por estes trabalhos serão apresentados abaixo.

2.1 DIAGRAMAS

Durante a realização do levantamento e análise de requisitos, são elaborados diagramas para expressar de forma gráfica a modelagem do sistema a ser desenvolvido. A UML é uma forma padrão de fazer os diagramas necessários para o desenvolvimento de software, De acordo com Booch; Rumbaugh; Jacobson (2006, p. 13):

> A UML é uma linguagem-padrão para a elaboração da estrutura de projetos de software... que poderá ser empregada para a visualização, a especificação, a construção e a documentação de artefatos que façam uso de sistemas complexos de software.

Dentro deste contexto dois diagramas foram selecionados para uso neste trabalho, o Diagrama Casos de Uso e o Diagrama Entidade-Relacionamento, pois através do uso destes é possível obter um padrão às estruturas que serão analisadas dentro do software Astah.

2.1.1 Diagrama Casos de Uso

O objetivo deste diagrama é permitir a compreensão do comportamento externo do sistema, utilizando uma linguagem simples, sendo acessível à seus usuários. Devido à linguagem, este é o diagrama mais abstrato da UML, este

apresenta o sistema por um perspectiva do usuário permitindo visualizar as principais funcionalidades do sistema.

O Diagrama Casos de Uso é de grande auxílio no processo de aquisição de requisitos, pois permite especificar, visualizar ou documentar características do futuro sistema. (GUEDES, 2009 p.55). Um exemplo com os elementos básicos é mostrado na Figura 1.

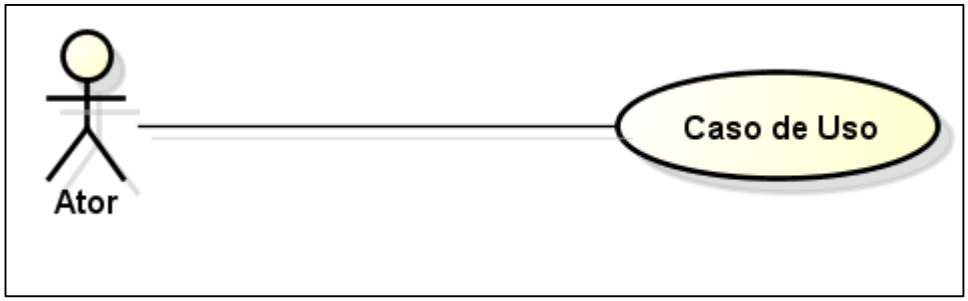

Figura 1 - Exemplo de Digrama Caso de Uso.
Fonte: Próprio Autor.

Os Atores e os Casos de Uso são os principais elementos deste diagrama, estes são utilizado para delimitar a fronteira do sistema, ou uma perspectiva dentro do sistema. Os elementos tem prioridade em mostrar as funcionalidades que serão abordadas ao longo do desenvolvimento sem necessariamente modelar a arquitetura do sistema, ou como os casos de uso serão implementados. A principal vantagem desse diagrama é a livre criação do esboço do sistema.

Os atores representam os papéis desempenhados pelos diversos usuários que poderão utilizar os serviços e funções do sistema (GUEDES, 2009 p.56). No Astah os atores são graficamente representado por “bonecos palito” ou retângulos marcados com a palavra ator abaixo ou no elemento gráfico. Sua função é representar os diversos usuários que o sistema poderá receber, sendo que não é obrigatório o ator ser propriamente uma pessoa, ou somente o cliente da aplicação, mas qualquer papel desempenhado por pessoa ou sistema que utilizará uma funcionalidade neste sistema que está sendo projetado.

Os casos de uso são utilizados para capturar os requisitos do sistema, que poderão ser utilizados pelos atores, este elemento é representado graficamente

por um balão nos programa de modelagem adotado, o Astah. Os casos de uso expressam e documentam os comportamentos esperados para determinada funcionalidade descrita, fornecendo instruções em linhas gerais de como serão realizadas sua usabilidade. O diagrama mostra através duma linguagem visual, as informações essenciais sem entrar em detalhes de implementação. Na Figura 1 por exemplo, o ator interage com o caso de uso. Este é usado para mostrar quais serão os cenários de interação, as restrições e outras informação que forem necessárias. É preciso ressaltar que esta documentação não possui um formato específico.

As interações ou relacionamentos são exibidos no diagrama usando linhas que ligam os atores aos casos de uso. Existem três tipos de associação: a inclusão, a extensão e a generalização. Essas recebem nomes específicos de acordo com o elemento ao qual estão ligadas e também podem variar o significado semântico. Abaixo é apresentado um exemplo de inclusão na Figura 2.

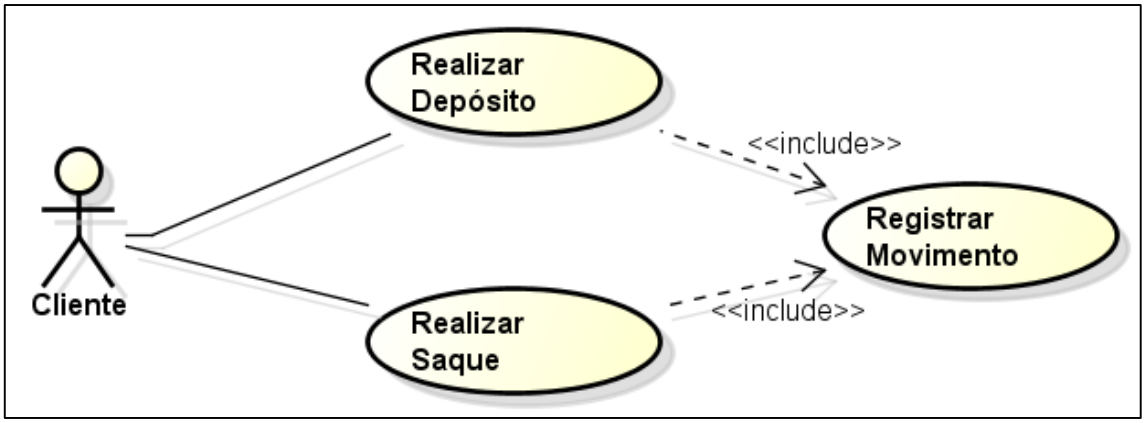

Figura 2 - Exemplo de Inclusão em um Diagrama de Caso de Uso.
Fonte: Retirado de Guedes (2009, p. 64).

Nesse exemplo, lê-se que, toda vez que o Cliente for realizar um depósito é necessário e obrigatório realizar um registro do movimento. Outra interpretação possível é, toda vez que for realizado um saque é necessário e obrigatório registrar o movimento. Na Figura 3 é apresentado uma associação onde não é obrigatório utilizar o caso de ligado, tornando este opcional.

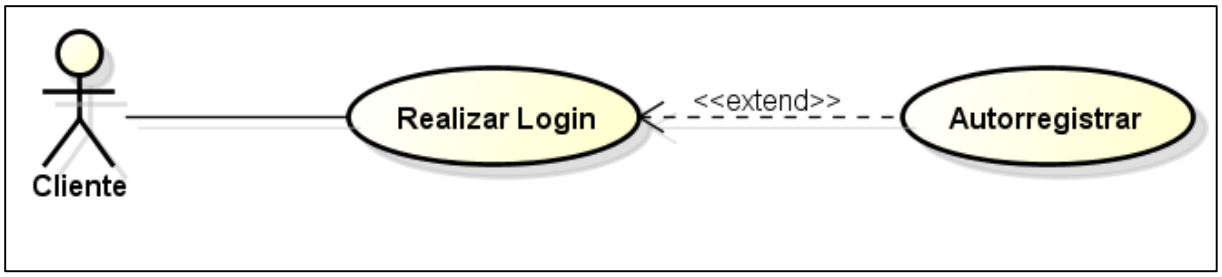

Figura 3 - Exemplo de Extensão em um Diagrama de Caso de Uso.
Fonte: Retirado de Guedes (2009, p. 66).

Na associação de Extensão é utilizado para descrever cenários opcionais de um caso de uso. Na Figura 3, se um cliente realizar login, esse pode também fazer o uso do recurso autoregistrar.

Segundo Guedes (2009, p.66) os Estereótipos podem atribuir funções extras ou diferentes a um componente, portanto estes permitem que sejam utilizados para modelar situações diferentes das quais inicialmente teriam sido projetados, se for preciso podem ser criados novos estereótipos para obter uma modelagem adequada.

2.1.2 Diagrama Entidade-Relacionamento

No Diagrama de Entidade-Relacionamento figuram como elementos as entidades, os atributos, os relacionamentos e a cardinalidade. As entidades representam um objeto único e possui seus próprios atributos. Esses atributos, representam as entidades e tem propriedades descritivas sobre elas. Cada atributo tem um domínio, que podem ser: simples ou compostos, monovalorados ou multivalorados, armazenados ou derivados, nulos, chaves ou determinantes.

Os relacionamentos são associações entre uma ou várias entidades, onde devem ser observadas as restrições e interpretações sobre estas ligações. Com essas associações a cardinalidade desempenha a função de qual é o número de instâncias que uma entidade pode participar (ELMASRI; NAVATHE, 2011). Esses elementos se relacionam fornecendo uma representação adequada para a visualização das informações, relacionamentos e suas restrições. De acordo com

Silberschatz; Korth; Sudarshan, os diagramas de entidade-relacionamento auxiliam no desenvolvimento da modelagem de representação de dados (SILBERSCHATZ; KORTH; SUDARSHAN, 2006). Para visualizar o diagrama, a Figura 4 apresenta uma modelagem feita no Astah Professional.

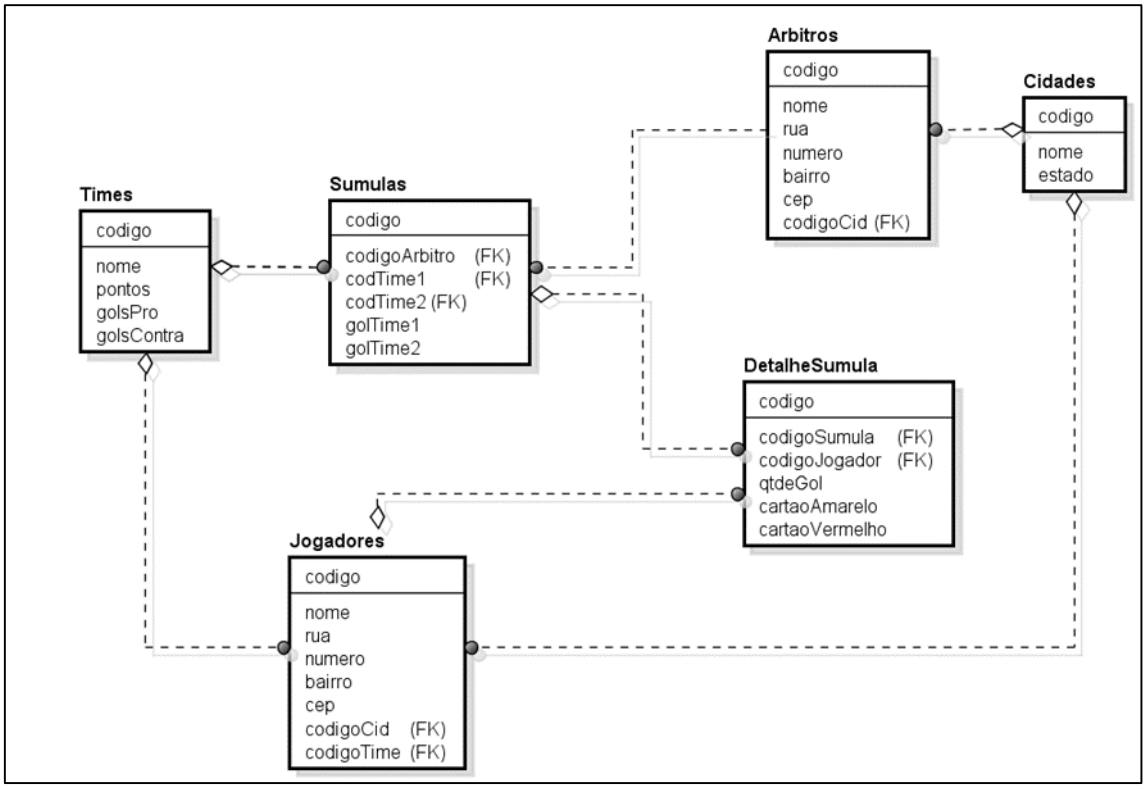

Figura 4 - Exemplo de Diagrama Entidade-Relacionamento (DER).
Fonte: Retirado de BLOG ENGENHARIA DE SOFTWARE, 2015.

Este diagrama representa um sistema para automatizar jogos de futebol, no exemplo acima temos as entidades, representadas pelos retângulos, e as linhas que representam o relacionamento as entidades. As entidades segundo são a representação gráfica de objetos no mundo real, sendo distinguível entre os outros objetos e possuindo algo que identificará este objeto unicamente, ou seja, uma chave primária. A chave primária é a responsável por identificar unicamente a entidade, como apresentado na figura acima os jogadores possuem um código que não se repetirá entre outras instâncias desses jogadores.

Além dos elementos mencionados existem também os atributos, que são cada um dos campos que guardarão logicamente as informações das entidades e

também existe outro tipo de chave, a chave estrangeira, identificada com a sigla “FK” que também poderá identificar as entidades e os atributos que essa entidade precisa de outra entidade indicando um relacionamento.

No exemplo mostrado pela Figura 4, os times agregam jogadores, sendo este um tipo de relacionamento, onde as entidades existem de forma independentes, um jogador pode existir e não estar agregado a um time, e um time pode existir e não ter agregado jogador algum, esse tipo de relação é mais facilmente identificada no modelo entidade-relacionamento, como forma de incluir isto nos diagramas entidade-relacionamento existe o losango não preenchido em uma das extremidades das linhas tracejadas. Quando o losango é preenchido, este representa outro relacionamento, composição, que indica que que uma entidade não existe se a entidade que a possui não existir. Também é possível incluir a cardinalidade indicada por números para identificar quantidade de entidades relacionadas a uma outra, ou então “P” para indicar pode existir várias entidades relacionadas (SILBERSCHATZ; KORTH; SUDARSHAN, 2006).

2.2 PONTOS POR FUNÇÃO

As métricas desempenham um papel de controle sobre os projetos, pois permitem quantizar esforços para produzir determinado produto. Os Pontos por Função foram desenvolvidos por Allan Albrecht em 1979, sendo depois refinados e melhorados pelo Grupo Internacional de Usuários de Pontos de Função (PRESSMAN, 1992). Esta técnica mede o tamanho funcional de softwares, sendo considerado uma medida de tamanho de software baseada em uma avaliação padronizada dos requisitos lógicos dos usuários. Segundo a IFPUG os Pontos de Função quantificam as funções contidas no software em termos que sejam significativos para os usuários do software, relacionando diretamente os requisitos de negócio de software com os objetivos do software (IFPUG, 2015). Nesta técnica existe duas classes de funções, as Funções de Dados e as Funções de Transação, na primeira procura-se atender requisitos de dados externos e internos, enquanto nas Funções de Transação tem o objetivo de atender funcionalidades da aplicação para o processamento desses dados (DEKKERS, 1998).

Antes de iniciar a contagem dos pontos é preciso estabelecer o que será contabilizado. De acordo com Mendes os pontos por função podem ser aplicados em projetos a serem desenvolvidos, na manutenção em um sistema, ou na determinação de sistemas existentes, não ficando restrito a apenas estes cenários. Também faz-se necessário saber o escopo do sistema, baseando-se em suas funcionalidades e não em sua implementação. Quando determinado a fronteira do sistema, cinco informações são relevantes para os pontos por função, Arquivo Lógico Interno (ALI), Arquivo de Interface Externa (AIE), Consulta Externa (CE), Entrada Externa (EE), Saída Externa (SE) como apresentado na Figura 5 (MENDES, 2014).

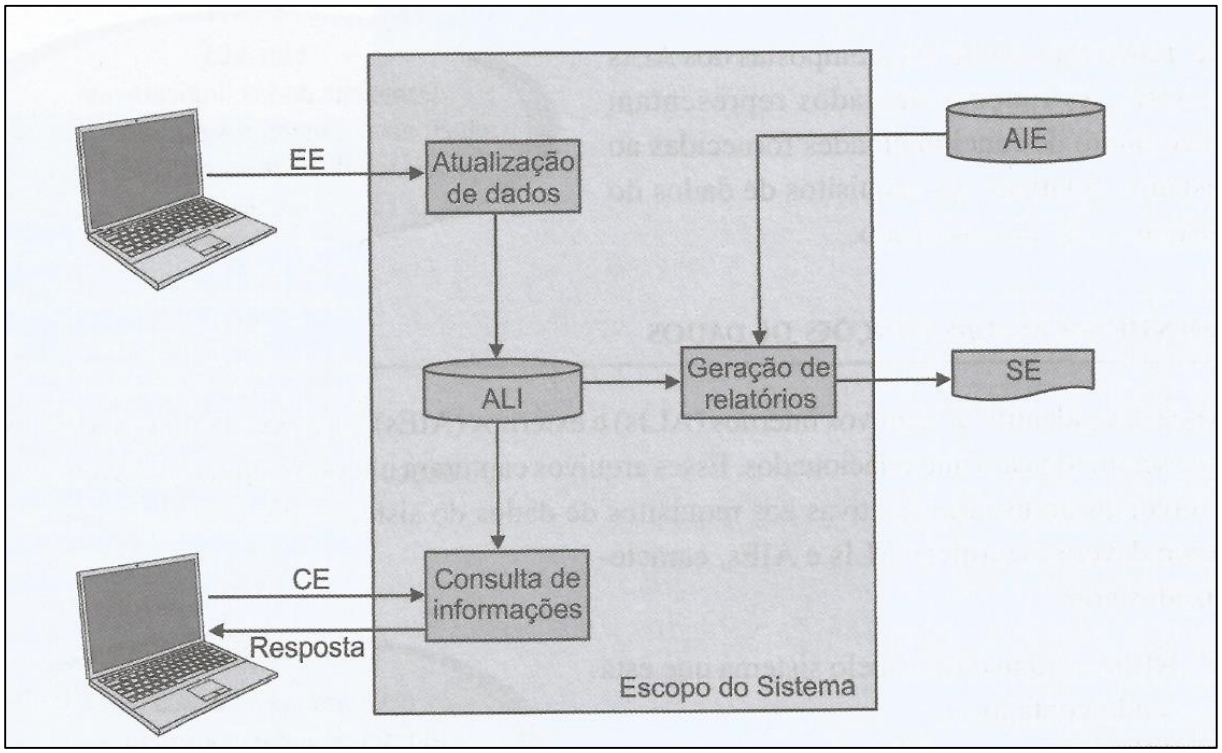

Figura 5 – Visão das Funções dentro de um Sistema.
Fonte: Retirado de Mendes (2014, p. 51).

2.2.1 Arquivo Lógico Interno (ALI)

Um ALI é uma entidade lógica e persistente, a respeito da qual dados serão mantidos. Estes se baseiam nos requisitos lógicos e serão encontrados nos diagramas de entidade-relacionamento (DEKKERS, 1998).

Para conta-los utiliza-se o diagrama mencionado e observa-se a quantidade de entidades, é considerado que cada entidade tem pelo menos um Registro Lógico Referenciado (RLR), que são conjunto de dados. Dentro de cada conjunto de dados existem subconjuntos de dados que possuem itens de dados (ID). Para exemplificar considere o computador sendo uma entidade, este é um RLR, que possue vários IDs como marca, modelo, fabricante, componentes, etc.

2.2.2 Arquivo de Interface Externa (AIE)

Um AIE é considerado uma entidade lógica e persistente, que será requisitada pela aplicação que está sendo desenvolvida para validação, logo um AIE pode é um ALI para outra aplicação que o mantém e é responsável por suas alterações.

2.2.3. Entrada Externa (EE)

A Entrada Externa funciona como um processo de onde grava informações em arquivos lógicos, ou os controla a fim de manter o software com seus dados devidamente atualizados. Um exemplo de EE, no caso de uma entidade computador, seria instalar um novo componente.

2.2.4 Saída Externa (SE)

Uma saída externa é um processo lógico que gera dados de saída, assim como a geração de um relatório ou mesmo o processo de gravação em um mídia, tem como principal objetivo gerar informações para um usuário.

2.2.5 Consulta Externa (CE)

Esta função é baseada em solicitação e resposta, logo aguarda uma requisição de serviço para obter dados externos a aplicação, os dados são recuperados para atender à solicitação e enviados para fora. Mais uma vez, as Consultas Externas fazem parte dos requisitos lógicos dos usuários de um aplicativo de software.

Cada tipo de função (ALI, AIE, EE, SE, CE) é avaliado, recebendo complexidade Baixa, Média ou Alta, utilizando um método direto que é delineado no Manual de Práticas de Contagem (IFPUG, 2015). O Quadro 1 resume os pesos que cada tipo de função pode receber.

Componente	Complexidade	Pesos
ALI	Baixa	7
	Média	10
	Alta	15
AIE	Baixa	5
	Média	7
	Alta	10
EE	Baixa	3
	Média	4
	Alta	6
SE	Baixa	4
	Média	5
	Alta	7
CE	Baixa	3
	Média	4
	Alta	6

Quadro 1 - Tabela de Pesos para Pontos por Função.
Fonte: Retirado de (FPACOCOMO, 2008, p. 1).

Para exemplificar a utilização do Quadro 1, será utilizada a Figura 6 que mostra um Diagrama de Casos de Uso de um Sistema Fictício de Redes.

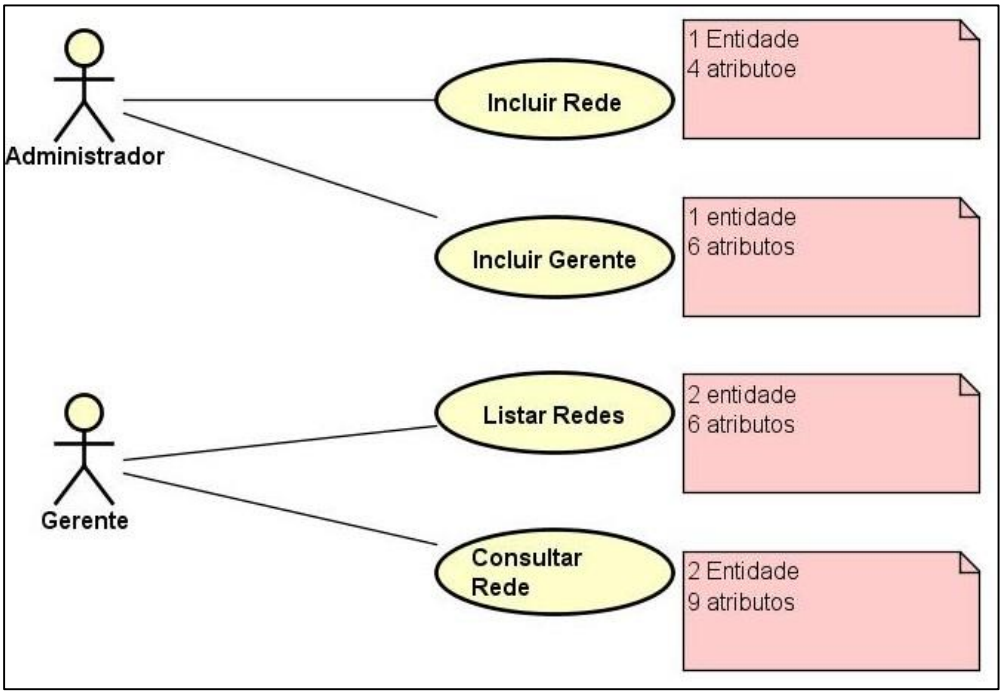

Figura 6 - Exemplo Contagem Pontos por Função.
Fonte Próprio Autor.

Nesse exemplo temos duas entradas externas, um saída externa e uma consulta externa. Nas notas ao lado de cada caso de uso temos as informações de quantas entidades são precisas acessar e a quantidade de campos que serão utilizados para a execução de suas funcionalidades. De acordo com o planilha digital (FPACOCOMO, 2008, p. 1) incluir uma rede é uma entrada externa que tem complexidade baixa, pois uma entidade equivale à um tipo de arquivo referenciado e possui 4 tipos de elementos de dados, logo no Quadro 1, observa-se que possui peso 3.

Para incluir um gerente manipula-se um tipo de arquivo referenciado e possui 6 tipos de elementos de dados, sendo classificado também como complexidade baixa, logo no Quadro 1, observa-se que possui peso 3.

Listar as redes é considerado uma saída externa e manipula dois tipos de arquivos referenciados e possui 6 tipos de elementos de dados, sendo classificado como complexidade média, logo no Quadro 1, observa-se que possui peso 6.

Já para consultar redes é considerado uma consulta externa, com dois tipos de arquivos referenciados e possui 9 tipos de elementos de dados, tendo complexidade média, no Quadro 1 apresenta peso 4. Logo este projeto totaliza 16 pontos por função.

3 IMPLEMENTAÇÃO

Ao longo deste capítulo serão apresentados os passos seguidos para o desenvolvimento do plug-in Function Points 1.0 para o Astah Professional, apresentando os métodos e procedimentos utilizados para obter a arquitetura do plug-in e suas funcionalidades.

3.1 MÉTODOS E PROCEDIMENTOS

Este trabalho foi realizado utilizando a metodologia de experimento controlado adaptado para software, no primeiro passo foi construído uma formulação, no caso de softwares faz-se necessário a construção dos Diagrama de Casos de Uso contendo as funcionalidades esperadas, nesta primeira etapa foi definido o problema a ser a abordado e o objetivo.

Na segunda etapa foi definido o tipo de projeto experimental, construção de um Plug-in para o Astah, nesta etapa foi estudada a teoria de Pontos por Função e explorado as possibilidades de uso para o pacote de desenvolvimento para o Astah, cedido pela Change Vision.

Na terceira etapa, deu-se início à implementação que é apresentada neste capítulo, por se tratar de um experimento controlado, protótipos foram construídos e testados pelos autores deste trabalho, ao final de cada teste de protótipo foram sugeridas melhorias e adições de novas partes da contagem de Pontos por Função, até a finalização do projeto e documentação neste trabalho.

3.2 ARQUITETURA DO PLUGIN

Para poder construir um plug-in para o Astah é necessário, algumas configurações iniciais, como baixar e instalar os programas sugeridos pelo tutorial de desenvolvimento de plug-ins do Astah (ASTAH TUTORIAL, 2013).

3.2.1 Configuração do Ambiente

Os programas utilizados para este desenvolvimento foram, o Sistema Operacional Windows 8.1 Pro com a licença de estudante cedida pela UTFPR, o Eclipse Luna com o plug-in do Maven (ECLIPSE LUNA, 2015), o Astah Professional com licença de estudante, pacote de desenvolvimento para o Astah (ASTAH SDK, 2015) e também a ferramenta de desenvolvimento da linguagem Java (JAVA SDK, 2015).

É necessário começar adicionando as duas variáveis de ambiente, ASDK_HOME e JAVA_HOME, fazendo com que estas apontem aos caminhos dos pacotes de desenvolvimento, e anexando as variáveis criadas à variável *path* que permite com que sejam utilizados os programas nos caminhos apontados através do *Prompt de Comando*.

Depois das configurações iniciais é usual realizar o teste dos comandos disponíveis no SDK do Astah e dar início a execução dos projetos. Os comandos utilizados podem ser visualizados na Quadro 1.

Comandos	Funções
asdk	Mostra a versão do Astah SDK
astah-generate-project	Criar um projeto novo de plug-in para o Astah
astah-build	Compila o projeto no caminho que o Prompt indica
astah-launch	Abre o software Astah
astah-mvn eclipse:eclipse	Indica para o projeto que a IDE que será utilizada é o Eclipse

Quadro 2 - Comandos do Astah SDK.
Fonte: Autor.

3.2.2 Arquitetura de Plug-ins no Astah

Na Figura 7 é mostrado a arquitetura utilizada para funcionamento de plug-ins na plataforma Astah.

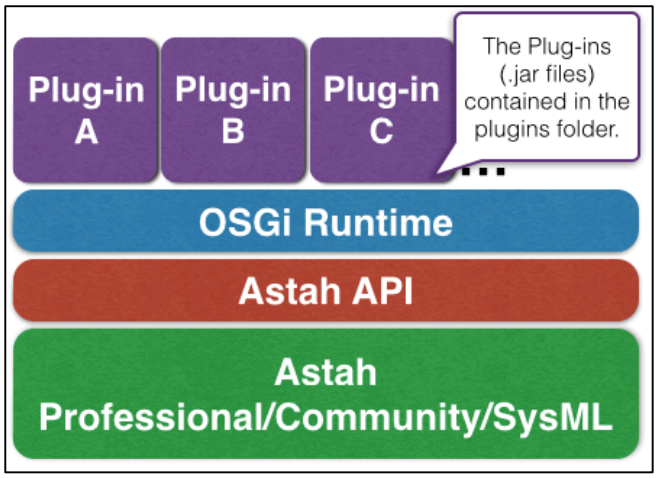

Figura 7 - Arquitetura Astah
Fonte: Retirado de Astah Tutorial, 2015.

Quando abre-se o Astah, as funcionalidades extras serão carregadas da pasta *plugins* onde o Astah foi instalado, os arquivos que estão dentro dessa pasta tem formato *.jar* e devem cumprir com o conjunto de especificações que definem um sistema dinâmico de componentes para Plataforma Java, o *OSGi Runtime* (OSGI, 2015), responsável pela comunicação com a API do Astah fazendo as chamadas das funções de interface do núcleo do Astah dependentes da versão que está sendo utilizada.

3.2.3 Arquitetura do Plug-ins Function Points 1.0

Na Figura 8 encontra-se o diagrama de Classes que contém as classes e os métodos desenvolvidos para o funcionamento do plug-in.

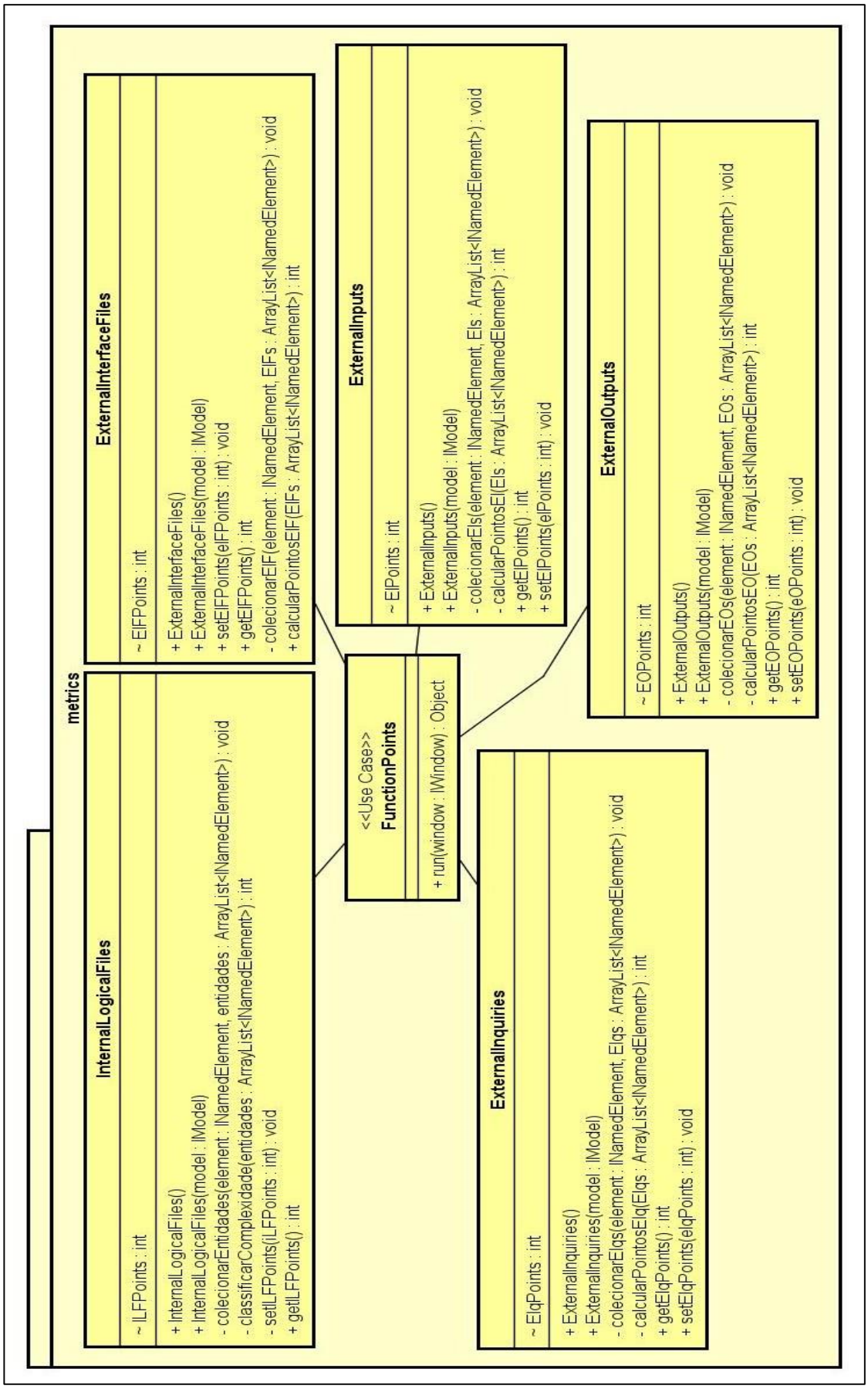

Figura 8 - Arquitetura FunctionPoints 1.0.
Fonte: Próprio Autor.

Na implementação cada uma das partes analisadas em Pontos por Função, foi considerada uma classe, sendo a principal a classe *FunctionPoints* que realiza a soma dos pontos contados nas demais classes.

Assim que uma das classes que conta pontos é instanciada pelo seu método construtor, um algoritmo é executado como padrão, primeiro é definido a variável que acumulará os pontos. Em seguida alguns métodos são chamados, os métodos de colecionar os elementos que serão posteriormente utilizados no método que calculará os pontos de acordo com as complexidades dos elementos analisados. No caso de Entidades só é necessário classificar as complexidades para obter os pontos e nos Casos de Uso com os estereótipos definidos (já classificados) são necessários calcular os pontos, esse diferenciação fica clara no diagrama acima.

O método *colecionarEntidades* é um método recursivo que procura Entidades em diagramas de Entidade-Relacionamento para inserir em uma lista, *ArrayList* (JAVA API, 2015) do tipo *IEREntity* (ASTAH API, 2015), que serão classificadas depois em *classificarComplexidade*. No método que coleciona, é preciso navegar em profundidade verificando e descartando elementos que são Pacotes, Modelos e Esquemas para encontrar e adicionar à lista as Entidades. Esse método é aplicado de forma similar em todas as classes que realizam a contagem, apenas adicionando a verificação de estereótipos ao invés de encontrar Entidades no diagrama de casos de Uso.

O método *classificarComplexidade* inicializa a variável de contagem em zero e verifica se a Entidade tem atributos, caso não haver, não haverá complexidade a ser considerada pois a quantidade de Tipos de Elementos de Dados é zero. Em uma Entidade o número de Tipos de Elemento de Registro é um, pois só há uma coluna que se referência a própria Entidade, diferente de um arquivo XML que tem várias *tags*, e por isso apresenta Tipos de Elemento de Registro igual ao número de *tags*. O Algoritmo passa a obter o número de atributos e fazer verificações se a complexidade é baixa, média ou alta, atribuindo os pesos 7, 10, 15 respectivamente a cada Entidade da lista.

O método calcular, se distingue em três letras no final para diferenciar a classe à qual pertence, porém funciona de forma similar nas classes em que é empregado, varrendo a lista de elementos colecionados e verificando o número de Arquivos relacionados e também o número de Campos envolvidos com os casos de

uso estereotipados com Entrada, Saída ou Consulta, posteriormente recebendo os pesos especificados no Quadro 1 (FPACOCOMO, 2008).

3.3 ADAPTAÇÕES NOS DIAGRAMAS

De acordo com o conceito apresentado sobre Casos de Uso, foi visto que é possível criar estereótipos que representem o papel adequado para a modelagem e também que é possível fazer a documentação de forma livre, pois a função deste diagrama é justamente mantê-lo simples, apresentando as principais funcionalidades como forma de limitar o escopo do sistema. No projeto que será desenvolvido será necessário criar estereótipos específicos para os atores, para representar corretamente a modelagem faz-se necessário atribuir um valor ao elemento contido no diagrama com a finalidade de somar os pontos referentes a este tipo de função, como no caso dos Arquivos de Interface Externa, onde será necessário criar um novo estereótipo.

Seguindo a mesma lógica, será necessário ter um valor associado aos elementos que estão no diagrama, para a classificação dos Casos de Uso pelo Plug-in, deverá ser criar um valor de etiqueta, chamado de TaggedValue no Astah. Esse valor poderá ser especificado com qualquer uma das etiquetas que representem as funções de transação em Pontos por Função, por exemplo, Entrada Externa (EE), Saída Externa (SE), Consulta Externa (CE). Para haver a correta contabilização dos Pontos por Função essas adaptações são essenciais e não fogem da realidade dos projetos, adicionando apenas informações que não interferirão sobre a modelagem e fazem parte da boa prática de documentação de projetos. No diagrama entidade-relacionamento não se faz necessário adaptações, pois como explicitado em Pontos por Função, os Arquivos Lógico Interno estão ligados as entidades que retém os dados, logo os pontos relacionados aos ALIs serão contabilizados sobre a quantidade de entidades e a quantidade de seus atributos.

3.4 GUIA DE UTILIZAÇÃO

Para instalar um Plug-in no Astah é necessário copiar o plug-in para a pasta "*plugins*", pasta que pode ser encontrada dentro do diretório que foi instalado o Astah, em seguida abra o programa Astah e encontre a guia "*Help*" e dentro dessa guia o menu "*Plugin List*". Verifique na tela que apareceu se o nome do plug-in que instalou aparece na lista.

Para a utilização do Plug-in Function Points 1.0 deve-se seguir a forma de documentar os casos de uso como detalhado no tópico anterior, estereotipando corretamente e adicionando as informações adicionais já descritas. Na ferramenta desenvolvida neste trabalho cada Entidade criada com atributos no Diagrama Entidade Relacionamento é contabilizado como um ALI e dependendo da quantidade de atributos recebe um nível de complexidade e consequentemente uma quantidade de pontos referente ao nível encontrado.

Na classificação das demais funções é preciso utilizar estereótipos, logo para indicar uma função de entrada, estereotipe um caso de uso com "*input*", uma função de saída com “*output*” e uma função de consulta com “*inquiry*” e para indicada um Arquivo de Interface Externa estereotipe um ator com "*extern interface*".

Nas funções de entrada, saída e consulta ainda é necessário adicionar a quantidade de arquivos (“*Files*”) ao qual cada função se relaciona e também a quantidade de campos (“*Fields*”) que devem ser acrescentados em "*TaggedValue*".

E depois de de documentar corretamente o projeto no qual está trabalhando vá até a guia “*Tools”* encontrando o menu “*Metrics*” e o submenu “*Function Points*”.

4 RESULTADOS

A ferramenta Function Points foi desenvolvida com uma abordagem mais prática e automática de totalizar os Pontos por Função, além de ser mais uma ferramenta para auxiliar engenheiros de software no controle de desenvolvimento de projetos de softwares, este trabalho tem como dividendo um guia que permite explorar de forma mais direta o desenvolvimento utilizando o Astah SDK, abrindo um horizonte de produção de novas ferramentas que realizem operações sobre os diagramas do software de modelagem, possibilitando que a comunidade acadêmica utilize os fundamentos aqui difundidos, aprimorando e produzindo mais.

4.1 EXEMPLO DE USO

Para este exemplo de uso deve-se seguir o que foi descrito no item 3.9. Para uma demonstração prática, serão descrito os passos para realizar o exemplo proposto abaixo. Para iniciar o usuário necessita abrir o Astah com o plugin já carregado e criar um diagrama de caso de uso, como mostra a Figura 9.

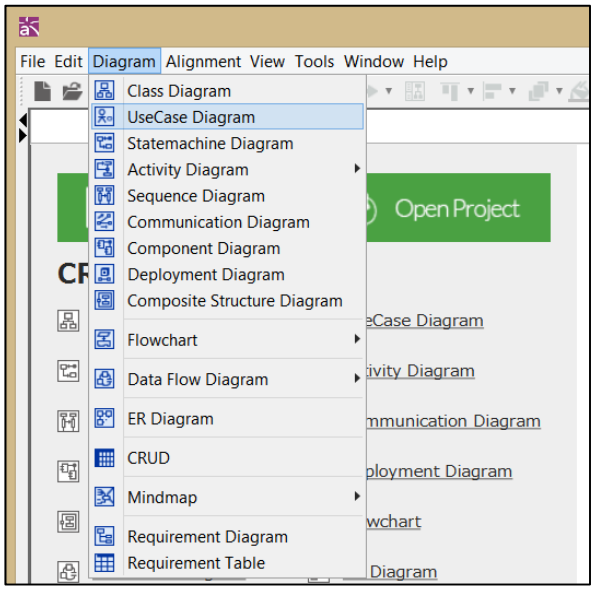

Figura 9 - Criando um Caso de Uso.
Fonte: Próprio Autor.

Após a criação do diagrama é necessário criar um ator (Engenheiro de Software) e um caso de uso (*Function Points*) e associá-los. No caso de uso deve-se adicionar um novo estereótipo (*input*) e selecioná-lo como representado na Figura 10.

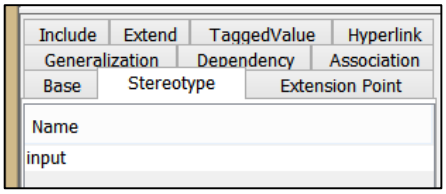

Figura 10 - Adicionando um estereótipo.
Fonte: Próprio Autor.

Ainda no caso de uso deve-se também adicionar os valores dois valores, a quantidade de arquivos (*Files*) e a quantidade(*Fields*) de campos que estão relacionados com este caso de uso. A frente desses valores coloca-se os valores correspondentes como mostrado na Figura 11.

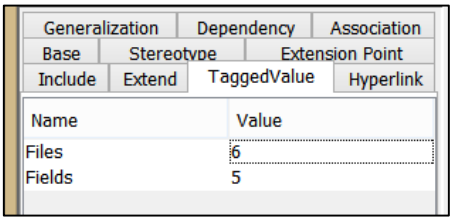

Figura 11 - Adicionando Valores de Etiqueta (TaggedValue).
Fonte: Próprio Autor.

Por último utiliza-se a função Function Points que está sob o menu Tools e sub-menu Metrics, obtendo os pontos que são apresentados na Figura 12.

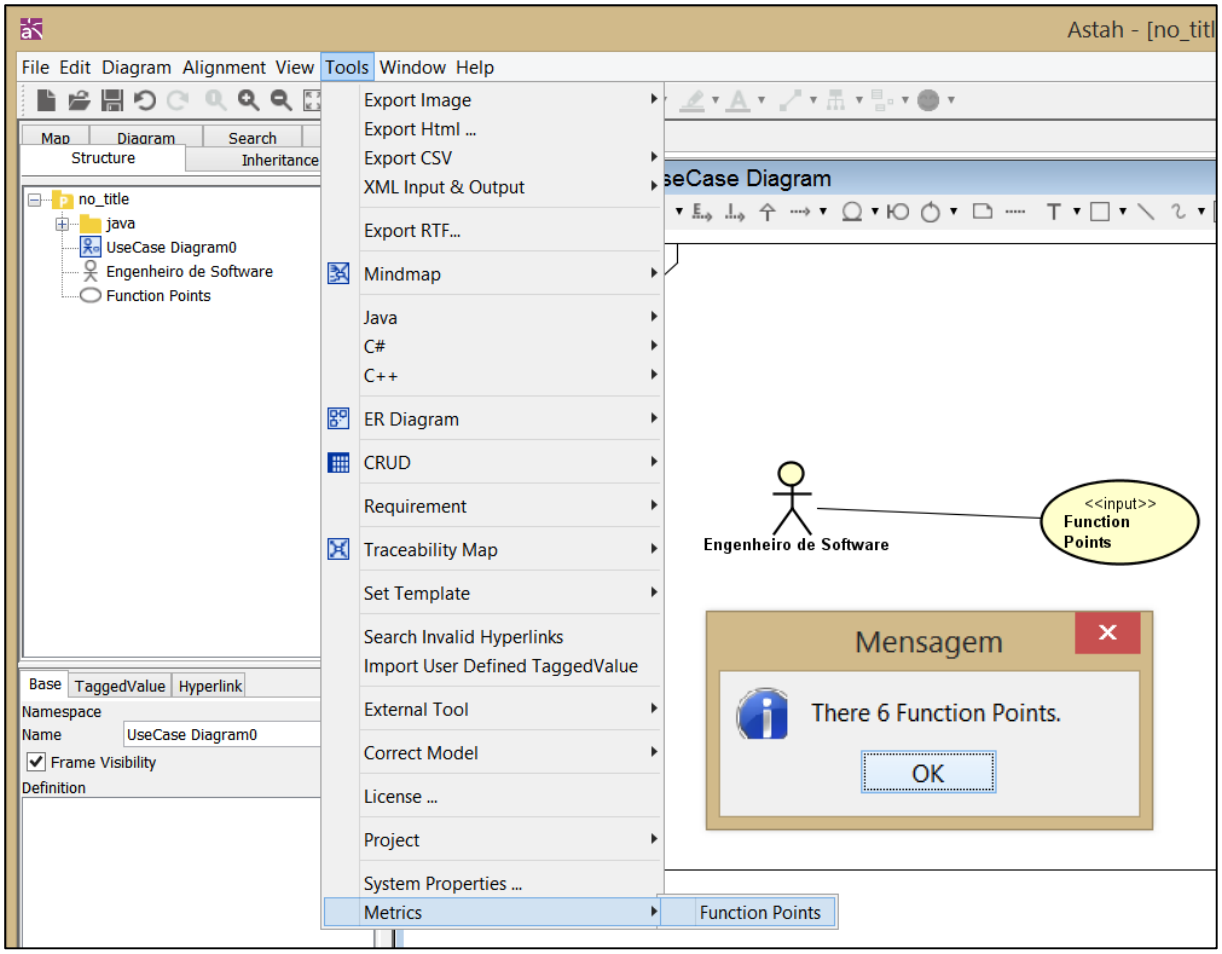

Figura 12 – Menu Function Points.
Fonte: Próprio Autor.

A Figura 13 apresenta o Caso de Uso utilizado para a modelagem inicial do plug-in que desenvolvido como exemplo nessa seção. A ferramenta está disponível no link[1] e ficará disponível também no site do Astah.

[1] https://www.dropbox.com/sh/pxxiyuuqvjvbpcv/AACiCZFwyUBEVgpj5D6EjZEDa?dl=0

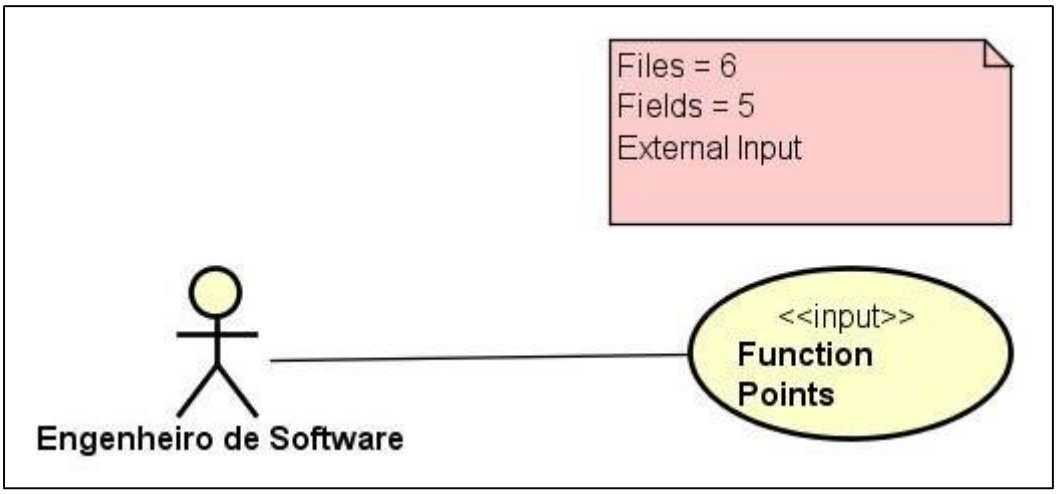

Figura 13 - Caso de Uso Plug-in FunctionPoints.
Fonte: Próprio Autor

Na Figura 13 foi explicitado a quantidade de arquivos que o caso de uso manipula e também a quantidade de Campos que ele acessa, por se tratar de uma entrada de dados no sistema desenvolvido é classificado como uma Entrada Externa, e de acordo com o Quadro 1, possui complexidade alta, contabilizando 6 pontos por função.

Já na Figura 14, mostra o Diagrama de Entidade-Relacionamento para o plug-in desenvolvido, tendo 6 entidades e cada uma dessas com um atributo apenas, sendo cada uma das entidade classificadas com complexidade baixa, recebendo 7 pontos cada de acordo com o Quadro 1, e totalizando 42 pontos por função.

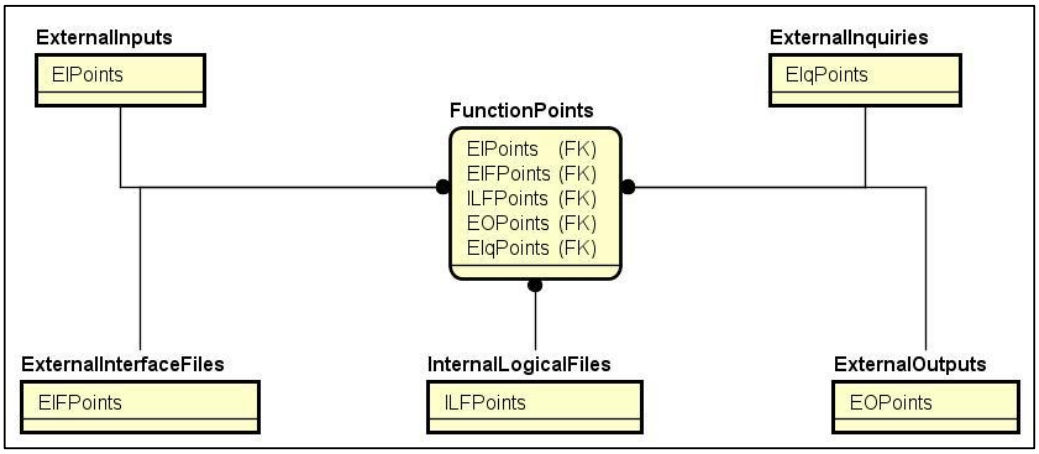

Figura 14 - DER Plug-in FunctionPoints.
Fonte: Próprio Autor

Somando os pontos adquiridos no projeto que contém os diagramas contabilizados, temos 48 pontos por função que são exibidos como resultado na Figura 15.

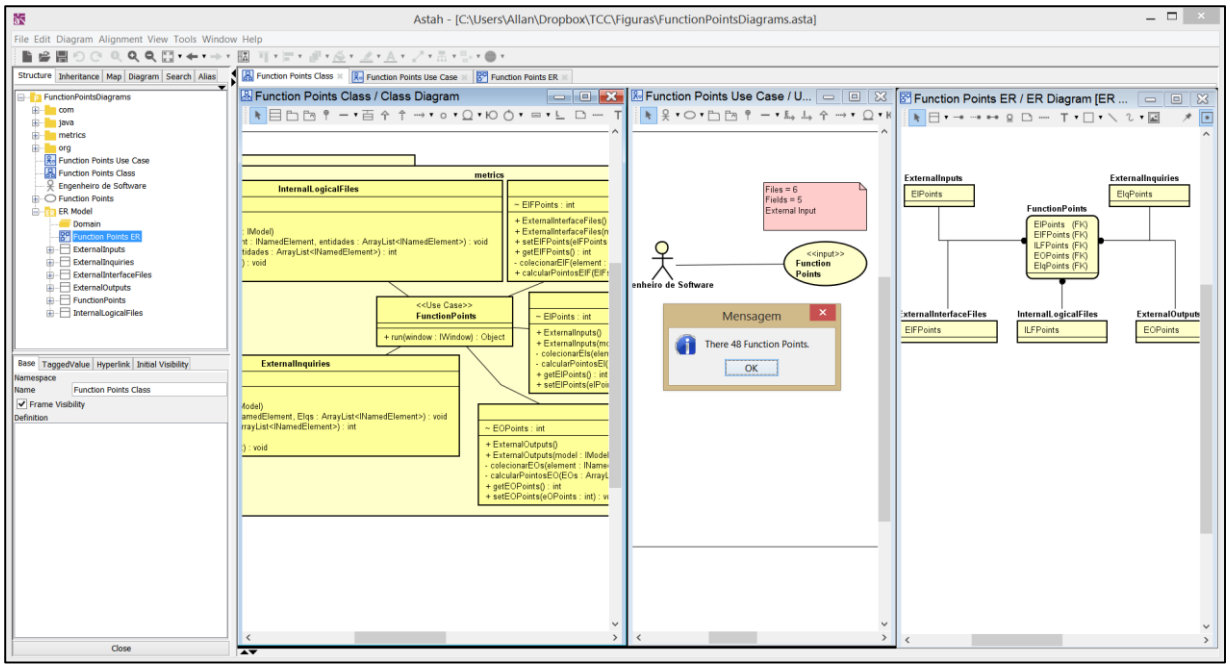

Figura 15 - Quantidade de Pontos por Função do Plug-in
Fonte: Próprio Autor.

Os diagramas de Entidade e Relacionamento e o Diagrama de Classe apresentados acima, foram gerados à partir dos código utilizados para o funcionamento do plug-in.

5 CONCLUSÃO

Este trabalho apresentou o desenvolvimento de um plug-in para o Astah Professional, o Function Points 1.0. Desde os conceitos de métricas de software, mostrando a implementação, passando pelo desenvolvimento e arquitetura até a demonstração de como utiliza-lo. A principal motivação para desenvolver esta nova funcionalidade foi devido ao cenário encontrado no item 1.1. Esse apresentava uma oportunidade para automatização de uma técnica manual de medir complexidade de funções em softwares. Tendo como resultado direto o aumento do controle e do gerenciamento de projetos de software, tornando mais prático a utilização de métricas.

Utilizando as ferramentas e tecnologias que possibilitaram o desenvolvimento do plugin no Astah, tornou-se possível automatizar a métrica de software, pontos por função, que pode beneficiar desenvolvedores e empresas preocupadas com desempenho em produção de aplicações e prazo de entrega de produtos. No capítulo de resultados algumas demonstrações da ferramenta são apresentadas, além de ser possível utilizar o plug-in, fazendo o download no link disponibilizado no item 4.1 e futuramente procurando na página oficial de plug-ins do Astah.

Com o desenvolvimento deste trabalho outras possibilidades surgem, como a melhoria deste plug-in para além de capturar os pontos por função, estes também possam ser armazenados num histórico de projetos e seus pontos já capturados. Outra possibilidade é adicionar outras métricas ao menu de métricas, complementando o código já desenvolvido. Uma outra contribuição é a possibilidade de utilizar os passos iniciais de desenvolvimento de plug-ins para o Astah e desenvolver novas funcionalidades e assim propagar o conhecimento concentrado nesse trabalho.

REFERÊNCIAS

ARGOUML. **ArgoUML Website,** 2015. Disponível em: <http://argouml.tigris.org/>. Acesso em: 07 abr 2015.

ASTAH API. **Astah API Website,** 2015. Disponível em: <http://astah.net/features/astah-api>. Acesso em: 07 abr 2015.

ASTAH. **Astah Website,** 2015. Disponível em: <http://astah.net/>. Acesso em: 07 abr 2015.

ASTAH SDK. **Astah SDK Website,** 2015. Disponível em: <http://astah.net/features/sdk>. Acesso em: 07 abr 2015.

ASTAH TUTORIAL. **Astah Tutorial Website,** 2015. Disponível em: <http://astah.net/tutorials/plug-ins/plugin_tutorial_en/html/_images/archtecture-3.png>. Acesso em: 07 abr 2015.

BLOG ENGENHARIA DE SOFTWARE. **Engenharia de Software wordpress,** 2015. Disponível em: < https://engenhariasoftware.wordpress.com/2012/04/28/curso-de-astah-professional-pos-utfpr/>. Acesso em: 06 abr 2015.

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. **UML**: guia do usuário. 2. ed. rev. e atual. Rio de Janeiro, RJ: Campus; Elsevier, 2006.

CHAOS 2013. **Chaos Manifesto 2013,** 2013. Disponível em: <http://www.versionone.com/assets/img/files/ChaosManifesto2013.pdf>. Acesso em: 31 mar. 2015.

DEKKERS, Carol A. **Pontos de Função e Medidas.** QAI Journal, Dez. 1998.

DIAS, Raquel. **Análise por Pontos de Função: Uma Técnica para Dimensionamento de Sistemas de Informação.** Brasilia, D.F. 2003.

DRACH, Marcos David. **Aplicabilidade de métricas por pontos de função em sistemas baseados em Web**. Campinas: S.P. 2005.

ECLIPSE LUNA. **Eclipse Luna Website,** 2015. Disponível em: <https://eclipse.org/luna/>. Acesso em: 3 abr. 2015.

ELMASRI, Ramez; NAVATHE, Shamkant B.. **Sistemas de Banco de Dados**. 6. ed. São Paulo: Addison Wesley, 2011.

GUEDES, Gilleanes T. A. **UML 2**: uma abordagem prática. São Paulo, SP: Novatec, 2009.

IFPUG. **Counting Practices Manual**. Version 4.2, 2004.

IFPUG. **International Function Point Users Group Website,** 2015. Disponível em: < http://www.ifpug.org/>. Acesso em: 14 abr. 2015.

FPACOCOMO. **Tabela para contabilizar Pontos por Função,** 2008. Disponível em: <http://dl.dropbox.com/u/3525445/fpaCocomo.xls>. Acesso em: 20 abr. 2015.

JAVA API. **Java API Website,** 2015. Disponível em: <http://docs.oracle.com/javase/7/docs/api/>. Acesso em: 03 abr. 2015.

JAVA. **Java Website,** 2015. Disponível em: <https://www.java.com/pt_BR/>. Acesso em: 03 abr. 2015.

JAVA SDK. **Java SDK Website,** 2015. Disponível em: <http://www.oracle.com/technetwork/pt/java/javase/downloads/index.html>. Acesso em: 03 abr. 2015.

JUNIOR, Marcelo Mazini Bomfim. **Repositório de Projetos de Software Mensurados em Pontos de Função**. Cornélio Procópio, PR. 2011.

MACEDO, Marcus Vinícius La Rocca. **Uma proposta de aplicação da métrica de pontos de função em aplicações de dispositivos portáteis**. Campinas, S.P. 2003.

MACORATTI. **Ferramentas UML OpenSource,** 2015. Disponível em: <http://www.macoratti.net/13/06/net_uml1.htm>. Acesso em: 28 mar. 2015.

MENDES, Antonio. **Custo de Software**: planejamento e gestão. 1. ed. Rio de Janeiro, RJ: Elsevier, 2014.

OSGI. **OSGI Website**, 2015. Dísponivel em: <http://www.osgi.org/Technology/HomePage>. Acesso em: 03 abr. 2015.

PAULA FILHO, Wilson de Pádua. **Engenharia de software**: fundamentos, métodos e padrões. 3. ed. Rio de Janeiro, RJ: LTC, 2009.

PRESSMAN, Roger S. **Software engineering**: a practitioner's approach . 3rd. ed. New York: McGraw-Hill, 1992.

SILBERSCHATZ, Abraham; KORTH, Henry F.; SUDARSHAN, S.. **Sistema de banco de dados**. 5. ed. São Paulo, SP: Makron, 2006.

STARUML. **StarUML Website,** 2015. Disponível em: <http://staruml.io>. Acesso em: 07 abr. 2015.

APÊNDICE A – Classe FunctionPoints

```
package metrics;

import javax.swing.JOptionPane;

import com.change_vision.jude.api.inf.AstahAPI;
import com.change_vision.jude.api.inf.exception.ProjectNotFoundException;
import com.change_vision.jude.api.inf.model.IModel;
import com.change_vision.jude.api.inf.project.ProjectAccessor;
import com.change_vision.jude.api.inf.ui.IPluginActionDelegate;
import com.change_vision.jude.api.inf.ui.IWindow;

public class FunctionPoints implements IPluginActionDelegate {

        public Object run(IWindow window) throws UnExpectedException {
            try {

                //Acesso do Projeto aberto
                AstahAPI api = AstahAPI.getAstahAPI();
                ProjectAccessor projectAccessor = api.getProjectAccessor();
                IModel model = projectAccessor.getProject();


                //Realiza a contagem
                int AllPoints = 0;

                InternalLogicalFiles ILFPoints = new InternalLogicalFiles(model);
                AllPoints = AllPoints + ILFPoints.getILFPoints();

                ExternalInterfaceFiles EIFPoints = new ExternalInterfaceFiles(model);
                AllPoints = AllPoints + EIFPoints.getEIFPoints();

                ExternalInputs EIPoints = new ExternalInputs(model);
                AllPoints = AllPoints + EIPoints.getEIPoints();

                ExternalOutputs EOPoints = new ExternalOutputs(model);
                AllPoints = AllPoints + EOPoints.getEOPoints();

                ExternalInquiries EIqPoints = new ExternalInquiries(model);
                AllPoints = AllPoints + EIqPoints.getEIqPoints();


                //Exibe os pontos obtidos
                JOptionPane.showMessageDialog(window.getParent(),"There " + AllPoints+ " Function
Points.");

            } catch (ProjectNotFoundException e) {
                String message = "Project is not opened. Please open the project or create new project.";
                            JOptionPane.showMessageDialog(window.getParent(), message, "Warning",
JOptionPane.WARNING_MESSAGE);
            } catch (Exception e) {
                    JOptionPane.showMessageDialog(window.getParent(), "Unexpected error has
occurred.", "Alert", JOptionPane.ERROR_MESSAGE);
                throw new UnExpectedException();
            }
            return null;
        }

}
```

APÊNDICE B – Classe InternalLogicalFiles

```
package metrics;

import java.util.ArrayList;

import com.change_vision.jude.api.inf.model.IERAttribute;
import com.change_vision.jude.api.inf.model.IEREntity;
import com.change_vision.jude.api.inf.model.IERModel;
import com.change_vision.jude.api.inf.model.IERSchema;
import com.change_vision.jude.api.inf.model.IModel;
import com.change_vision.jude.api.inf.model.INamedElement;
import com.change_vision.jude.api.inf.model.IPackage;

public class InternalLogicalFiles {

	int ILFPoints;

	public InternalLogicalFiles() {
		setILFPoints(0);
	}

	public InternalLogicalFiles(IModel model) {

		setILFPoints(0);

		//Gera os pontos
		ArrayList<INamedElement> entidades = new ArrayList<INamedElement>();
		colecionarEntidades(model,entidades);

		//Seta os pontos
		int points = classificarComplexidade(entidades);
		setILFPoints(points);

	}

	//Metodo Para poder encontrar todas entidades, mesmo que estejam dentro de outros
pacotes (Metodo Recursivo).
	private void colecionarEntidades(INamedElement element, ArrayList<INamedElement>
entidades) {

		if (element instanceof IPackage) {

			for(INamedElement ownedNamedElement :
((IPackage)element).getOwnedElements()) {
		colecionarEntidades(ownedNamedElement, entidades);
	}

      } else {

	if (element instanceof IERModel) {

		for(INamedElement ownedNamedElement : ((IERModel)element).getSchemata()) {
		  colecionarEntidades(ownedNamedElement, entidades);
		}

	} else {

		if(element instanceof IERSchema){

			for(INamedElement ownedNamedElement :
((IERSchema)element).getEntities()) {
			  colecionarEntidades(ownedNamedElement, entidades);
```

```
                }

                } else{

                        if(element instanceof IEREntity){

                                entidades.add((IEREntity)element);

                }

                }

        }

    }

        }

        //Classifica Complexidade de todos elementos, o total de pontos desta modalidade
        private int classificarComplexidade(ArrayList<INamedElement> entidades) {

                int PointsF=0;

                for(INamedElement entidade : entidades){

                        int Points = 0;
                        //Entidades possuem apenas um elemento de registro, ou seja, uma coluna
                        int TER = 1;

                        //Sem atributos nao ha complexidade a ser considerada
                        int TED = 0;

                        IERAttribute At1[] = ((IEREntity)entidade).getPrimaryKeys();
                        IERAttribute At2[] = ((IEREntity)entidade).getForeignKeys();
                        IERAttribute At3[] = ((IEREntity)entidade).getNonPrimaryKeys();

                        //Chaves estrangeiras sao contadas pelo numero de conexoes (2), por isso
deve ser divida pelo numero de conexoes (2)
                        TED = (At1.length) + ((At2.length) / 2) + (At3.length);

                        if ((TER > 0) && (TED > 0)) {

                                // Definindo Pesos

                                Points = 1;

                                if (TER == 1) {
                                        if (TED < 51) {
                                                Points = Points * 7;// baixa
                                        } else {
                                                Points = Points * 10;// media
                                        }
                                } else {
                                        if (TER == 2) {
                                                if (TED < 20) {
                                                        Points = Points * 7;// baixa
                                                } else {
                                                        if (TED < 51) {
                                                                Points = Points * 10;// media
                                                        } else {
                                                                Points = Points * 15;// alta
```

```
                                        }
                                    }
                                } else {
                                    if (TER == 3) {
                                        if (TED < 20) {
                                            Points = Points * 10;// media
                                        } else {
                                            Points = Points * 15;// alta
                                        }
                                    }
                                }
                            }

                        } else {
                            Points = 0;
                        }

                        PointsF = PointsF + Points;

                    }

                    return PointsF;

                }

                //Define o valor dos pontos
                private void setILFPoints(int iLFPoints) {
                    ILFPoints = iLFPoints;
                }

                //Obtem a quantidade de pontos
                public int getILFPoints() {
                    return ILFPoints;
                }

            }
```

APÊNDICE C – Classe ExternalInterfaceFiles

```
package metrics;

import java.util.ArrayList;

import com.change_vision.jude.api.inf.model.IClass;
import com.change_vision.jude.api.inf.model.IModel;
import com.change_vision.jude.api.inf.model.INamedElement;
import com.change_vision.jude.api.inf.model.IPackage;

public class ExternalInterfaceFiles {

	int EIFPoints;


	public ExternalInterfaceFiles() {
		setEIFPoints(0);
	}

	public ExternalInterfaceFiles(IModel model) {
		setEIFPoints(0);

		//Gera os pontos
		ArrayList<INamedElement> EIFs = new ArrayList<INamedElement>();
		colecionarEIF(model,EIFs);

		//Gera e Seta os pontos
		int points = calcularPointosEIF(EIFs);
		setEIFPoints(points);

	}

	public void setEIFPoints(int eIFPoints) {
		EIFPoints = eIFPoints;
	}

	public int getEIFPoints() {
		return EIFPoints;
	}

	//Metodo Para poder encontrar todas EIFs, mesmo que estejam dentro de
outros pacotes (Metodo Recursivo).
	private void colecionarEIF(INamedElement element,
ArrayList<INamedElement> EIFs) {

		if (element instanceof IPackage) {

			for(INamedElement ownedNamedElement :
((IPackage)element).getOwnedElements()) {
			colecionarEIF(ownedNamedElement, EIFs);
		}

	    } else {

		if (element instanceof IClass) {

			String[] Stereotype = element.getStereotypes();
				String Normal = null;

				for (int j = 0; j < Stereotype.length; j++) {
					Normal = Stereotype[j];
					String StereotypeBase = "extern interface";
```

```
                        if (Normal.equalsIgnoreCase(StereotypeBase)) {
                              EIFs.add((IClass)element);
                        }
                  }

            }

        }

      }

      public int calcularPointosEIF(ArrayList<INamedElement> EIFs) {

            //Variaveis
                  int PointsT=0;
                  int Points = 1;


                  for(INamedElement i   : EIFs){


                                          int TER=0;
                                          int TED=0;

                                          try{
                                                Integer Valor = new
Integer(i.getTaggedValue("RET"));
                                                TER = Valor.intValue();
                                                System.out.println("TER:
"+TER);
                                          }catch(Exception e){
                                          }


                                          try{
                                                Integer Valor = new
Integer(i.getTaggedValue("ret"));
                                                TER = Valor.intValue();
                                                System.out.println("TER:
"+TER);
                                          }catch(Exception e){
                                          }


                                          try{
                                                Integer Valor = new
Integer(i.getTaggedValue("DET"));
                                                TED = Valor.intValue();
                                                System.out.println("TED:
"+TED);
                                          }catch(Exception e){
                                          }


                                          try{
                                                Integer Valor = new
Integer(i.getTaggedValue("det"));
                                                TED = Valor.intValue();
                                                System.out.println("TED:
"+TED);
                                          }catch(Exception e){
                                          }
```

```
try {
      if ((TER > 0) && (TED > 0)) {
            // Definindo Pesos
            if (TER == 1) {
                  if (TED < 51) {
                        Points = Points * 5;// baixa
                  } else {
                        Points = Points * 7;// media
                  }
            } else {
                  if (TER == 2) {
                        if (TED < 20) {
                              Points = Points * 5;// baixa
                        } else {
                              Points = Points * 7;// media
                        }
                  } else {
                        if (TER == 3) {
                              if (TED < 20) {
                                    Points = Points * 7;// media
                              } else {
                                    Points = Points * 10;// alta
                              }
                        } else {
                              if (TED < 51) {
                                    Points = Points * 7;// media
                              } else {
                                    Points = Points * 10;// alta
                              }
                        }
                  }
            }
      } else {
            Points = 5;
      }

      PointsT = PointsT + Points;

      Points=1;

} catch (Exception E) {
```

```
                                                    // Excecao para nao
atributos
                                                }

                                    }

                            return PointsT;
                    }
}
```

APÊNDICE D – Classe ExternalInput

```
package metrics;

import java.util.ArrayList;

import com.change_vision.jude.api.inf.model.IClass;
import com.change_vision.jude.api.inf.model.IModel;
import com.change_vision.jude.api.inf.model.INamedElement;
import com.change_vision.jude.api.inf.model.IPackage;

public class ExternalInputs {

	int EIPoints;

	public ExternalInputs() {
		setEIPoints(0);
	}

	public ExternalInputs(IModel model) {
		setEIPoints(0);

		//Gera os pontos
		ArrayList<INamedElement> EIs = new ArrayList<INamedElement>();
		colecionarEIs(model,EIs);


		int points = calcularPointosEI(EIs);
		setEIPoints(points);

	}

	//Metodo Para poder encontrar todas EIs, mesmo que estejam dentro de
outros pacotes (Metodo Recursivo).
		private void colecionarEIs(INamedElement element,
ArrayList<INamedElement> EIs) {

			if (element instanceof IPackage) {

				for(INamedElement ownedNamedElement :
((IPackage)element).getOwnedElements()) {
					colecionarEIs(ownedNamedElement, EIs);
			}

		} else {

			if (element instanceof IClass) {

				String[] Stereotype = element.getStereotypes();
					String Normal = null;

					for (int j = 0; j < Stereotype.length; j++) {
						Normal = Stereotype[j];
						String StereotypeBase = "input";

						if
(Normal.equalsIgnoreCase(StereotypeBase)) {
							EIs.add((IClass)element);
						}
					}

			}

		}
```

```
		}

	private int calcularPointosEI(ArrayList<INamedElement> EIs) {
		//Variaveis
		int PointsT=0;
		int Points = 1;


		for(INamedElement i  : EIs){


					int Arquivos=0;
					int Campos=0;

					try{
						Integer Valor = new
Integer(i.getTaggedValue("Files"));
						Arquivos = Valor.intValue();
						System.out.println("Arquivos:
"+Arquivos);
					}catch(Exception e){
					}

					try{
						Integer Valor = new
Integer(i.getTaggedValue("files"));
						Arquivos = Valor.intValue();
						System.out.println("Arquivos:
"+Arquivos);
					}catch(Exception e){
					}

					try{
						Integer Valor = new
Integer(i.getTaggedValue("Fields"));
						Campos = Valor.intValue();
						System.out.println("Campos: "+Campos);
					}catch(Exception e){
					}

					try{
						Integer Valor = new
Integer(i.getTaggedValue("fields"));
						Campos = Valor.intValue();
						System.out.println("Campos: "+Campos);
					}catch(Exception e){
					}


						try {
							if((Arquivos == 0) || (Arquivos ==
1)){

								if(Campos >15){
									Points = Points * 4;//
media

								}else{
									Points = Points * 3;//
baixa

								}
							}else{
								if((Arquivos == 2) ||
```

```
(Arquivos ==3)){
                                                                  if(Campos<5){
                                                                        Points = Points
* 3;// baixa
                                                                  }else{
                                                                        if(Campos<=15){
                                                                              Points =
Points * 4;// media
                                                                        }else{
                                                                              Points =
Points * 6;// alta
                                                                        }
                                                                  }

                                                            }else{
                                                                  if(Arquivos > 3){
                                                                        if(Campos < 5){
                                                                              Points =
Points * 4;// media
                                                                        }else{
                                                                              Points =
Points * 6;// alta
                                                                        }
                                                                  }
                                                            }
                                                      }

                                                      PointsT = PointsT + Points;
                                                      Points=1;

                                                } catch (Exception E) {
                                                      // Excecao para nao atributos
                                                }

                              }

            return PointsT;
}


      public int getEIPoints() {
            return EIPoints;
      }

      public void setEIPoints(int eIPoints) {
            EIPoints = eIPoints;
      }


}
```

APÊNDICE E – Classe ExternalOutput

```
package metrics;

import java.util.ArrayList;

import com.change_vision.jude.api.inf.model.IClass;
import com.change_vision.jude.api.inf.model.IModel;
import com.change_vision.jude.api.inf.model.INamedElement;
import com.change_vision.jude.api.inf.model.IPackage;

public class ExternalOutputs {

	int EOPoints;

	public ExternalOutputs() {
		setEOPoints(0);
	}

	public ExternalOutputs(IModel model) {
		setEOPoints(0);

		//Gera os pontos
		ArrayList<INamedElement> EOs = new ArrayList<INamedElement>();
		colecionarEOs(model,EOs);


		int points = calcularPointosEO(EOs);
		setEOPoints(points);



	}
	//Metodo Para poder encontrar todas EIs, mesmo que estejam dentro de outros pacotes (Metodo Recursivo).
			private void colecionarEOs(INamedElement element, ArrayList<INamedElement> EOs) {

				if (element instanceof IPackage) {

					for(INamedElement ownedNamedElement : ((IPackage)element).getOwnedElements()) {
					colecionarEOs(ownedNamedElement, EOs);
				}

			} else {

				if (element instanceof IClass) {

					String[] Stereotype = element.getStereotypes();
						String Normal = null;

						for (int j = 0; j < Stereotype.length; j++) {
							Normal = Stereotype[j];
							String StereotypeBase = "output";

							if (Normal.equalsIgnoreCase(StereotypeBase)) {
								EOs.add((IClass)element);
							}
						}

				}
```

```
        }

    }

    private int calcularPointosEO(ArrayList<INamedElement> EOs) {
    //Variaveis
    int PointsT=0;
    int Points = 1;

    for(INamedElement i  : EOs){

                int Arquivos=0;
                int Campos=0;

                try{
                        Integer Valor = new
Integer(i.getTaggedValue("Files"));
                        Arquivos = Valor.intValue();
                        System.out.println("Arquivos: "+Arquivos);
                }catch(Exception e){
                }

                try{
                        Integer Valor = new
Integer(i.getTaggedValue("files"));
                        Arquivos = Valor.intValue();
                        System.out.println("Arquivos: "+Arquivos);
                }catch(Exception e){
                }

                try{
                        Integer Valor = new
Integer(i.getTaggedValue("Fields"));
                        Campos = Valor.intValue();
                        System.out.println("Campos: "+Campos);
                }catch(Exception e){
                }

                try{
                        Integer Valor = new
Integer(i.getTaggedValue("fields"));
                        Campos = Valor.intValue();
                        System.out.println("Campos: "+Campos);
                }catch(Exception e){
                }

                        try {
                                if ((Arquivos > 1) && (Campos > 1)) {
                                        // Definindo Pesos
                                                if ((Arquivos == 2) || (Arquivos
== 3) ) {

                                                        if ((Campos>=1) ||
(Campos <=5) ) {

                                                                Points =
Points * 4;// baixa

                                                        } else {
                                                                if
((Campos>=6) || (Campos <=19)) {

                                                                        Points
= Points * 5;// media
```

```
                                        } else {
                                                Points
= Points * 7;// alta
                                        }
                                }
                        } else {
                                if (Arquivos > 3) {
                                        if
((Campos>=1) || (Campos <=5)) {
                                                Points
= Points * 5;// media
                                        } else {
                                                Points
= Points * 7;// alta
                                        }
                                }
                        }

                } else {

                        if(Campos>19){
                                Points = 5;
                        }else{
                                Points = 4;
                        }

                }


                PointsT = PointsT + Points;
                Points=1;

        } catch (Exception E) {
                // Excecao para nao atributos
        }


                }
        return PointsT;
    }

    public int getEOPoints() {
        return EOPoints;
    }

    public void setEOPoints(int eOPoints) {
        EOPoints = eOPoints;
    }


}
```

APÊNDICE F – Classe ExternalInquiries

```
package metrics;

import java.util.ArrayList;

import com.change_vision.jude.api.inf.model.IClass;
import com.change_vision.jude.api.inf.model.IModel;
import com.change_vision.jude.api.inf.model.INamedElement;
import com.change_vision.jude.api.inf.model.IPackage;

public class ExternalInquiries {

	int EIqPoints;

	public ExternalInquiries() {
		setEIqPoints(0);
	}

	public ExternalInquiries(IModel model) {
		setEIqPoints(0);


		//Gera os pontos
		ArrayList<INamedElement> EIqs = new ArrayList<INamedElement>();
		colecionarEIqs(model,EIqs);


		int points = calcularPointosEIq(EIqs);
		setEIqPoints(points);


	}

	//Metodo Para poder encontrar todas EIqs, mesmo que estejam dentro de outros pacotes (Metodo Recursivo).
			private void colecionarEIqs(INamedElement element, ArrayList<INamedElement> EIqs) {

				if (element instanceof IPackage) {

					for(INamedElement ownedNamedElement : ((IPackage)element).getOwnedElements()) {
					colecionarEIqs(ownedNamedElement, EIqs);
				}

			} else {

				if (element instanceof IClass) {

					String[] Stereotype = element.getStereotypes();
						String Normal = null;

						for (int j = 0; j < Stereotype.length; j++) {
							Normal = Stereotype[j];
							String StereotypeBase = "inquiry";

							if (Normal.equalsIgnoreCase(StereotypeBase)) {
								EIqs.add((IClass)element);
							}
						}
```

```
            }

        }

    }

    private int calcularPointosEIq(ArrayList<INamedElement> EIqs) {
    //Variaveis

    int PointsT=0;
    int Points = 1;


    for(INamedElement i  : EIqs){

                int Arquivos=0;
                int Campos=0;

                try{
                        Integer Valor = new
Integer(i.getTaggedValue("Files"));
                        Arquivos = Valor.intValue();
                        System.out.println("Arquivos: "+Arquivos);
                }catch(Exception e){
                }

                try{
                        Integer Valor = new
Integer(i.getTaggedValue("files"));
                        Arquivos = Valor.intValue();
                        System.out.println("Arquivos: "+Arquivos);
                }catch(Exception e){
                }

                try{
                        Integer Valor = new
Integer(i.getTaggedValue("Fields"));
                        Campos = Valor.intValue();
                        System.out.println("Campos: "+Campos);
                }catch(Exception e){
                }

                try{
                        Integer Valor = new
Integer(i.getTaggedValue("fields"));
                        Campos = Valor.intValue();
                        System.out.println("Campos: "+Campos);
                }catch(Exception e){
                }

                        try {
                                if ((Arquivos > 1) && (Campos > 1)) {
                                        // Definindo Pesos
                                                if ((Arquivos == 2) || (Arquivos
== 3) ) {

                                                        if ((Campos>=1) ||
(Campos <=4) ) {

                                                                Points =
Points * 3;// baixa

                                                        } else {
```

```
                                                            if
((Campos>=5) || (Campos <=15)) {
                                                                Points
= Points * 4;// media
                                                            } else {
                                                                Points
= Points * 6;// alta
                                                            }
                                                        }
                                                    } else {
                                                        if (Arquivos > 3) {
                                                            if
((Campos>=1) || (Campos <=4)) {
                                                                Points
= Points * 4;// media
                                                            } else {
                                                                Points
= Points * 6;// alta
                                                            }
                                                        }
                                                    }

                                        } else {
                                            //Nenhum Arquivo referenciado ? 15
campos
                                            if(Campos<=15){
                                                Points = 3;//Baixa
                                            }else{
                                                Points = 4;//Media
                                            }

                                        }

                                        PointsT = PointsT + Points;
                                        Points=1;

                                    } catch (Exception E) {
                                        // Excecao para nao atributos
                                    }

                            }

            return PointsT;
        }

        public int getElqPoints() {
            return ElqPoints;
        }

        public void setElqPoints(int elqPoints) {
            ElqPoints = elqPoints;
        }

}
```

APÊNDICE G – Classe ExternalInterfaceFiles

```
package metrics;

import java.util.ArrayList;

import com.change_vision.jude.api.inf.model.IAttribute;
import com.change_vision.jude.api.inf.model.IClass;
import com.change_vision.jude.api.inf.model.IModel;
import com.change_vision.jude.api.inf.model.INamedElement;
import com.change_vision.jude.api.inf.model.IPackage;

public class ExternalInterfaceFiles {

	int EIFPoints;


	public ExternalInterfaceFiles() {
		setEIFPoints(0);
	}

	public ExternalInterfaceFiles(IModel model) {
		setEIFPoints(0);

		//Gera os pontos
		ArrayList<INamedElement> EIFs = new ArrayList<INamedElement>();
		colecionarEIF(model,EIFs);

		//Gera e Seta os pontos
		int points = calcularPointosEIF(EIFs);
		setEIFPoints(points);

	}

	public void setEIFPoints(int eIFPoints) {
		EIFPoints = eIFPoints;
	}

	public int getEIFPoints() {
		return EIFPoints;
	}

	//Metodo Para poder encontrar todas EIFs, mesmo que estejam dentro de outros pacotes
(Metodo Recursivo).
	private void colecionarEIF(INamedElement element, ArrayList<INamedElement> EIFs) {

		if (element instanceof IPackage) {

			for(INamedElement ownedNamedElement :
((IPackage)element).getOwnedElements()) {
		colecionarEIF(ownedNamedElement, EIFs);
	}

      } else {

	if (element instanceof IClass) {

		String[] Stereotype = element.getStereotypes();
				String Normal = null;

				for (int j = 0; j < Stereotype.length; j++) {
					Normal = Stereotype[j];
					String StereotypeBase = "extern interface";
```

```
                                        if (Normal.equalsIgnoreCase(StereotypeBase)) {
                                                EIFs.add((IClass)element);
                                        }
                                }

        }

    }

}

        public int calcularPointosEIF(ArrayList<INamedElement> EIFs) {

                        //Variaveis
                        int PointsT=0;
                        int Points = 1;
                        int TER = 0;

                        for(INamedElement i  : EIFs){

                                                TER++;
                                                int TED=0;


                                                        IAttribute atributos[] = ((IClass) i)
                                                                        .getAttributes();

                                                        try {
                                                                TED = atributos.length;
                                                                // TED quantidade de atributos
                                                                // TER quantidade de atores
estereotipados como interfaces externas

                                                                if ((TER > 0) && (TED > 0)) {
                                                                        // Definindo Pesos

                                                                        if (TER == 1) {
                                                                                if (TED < 51) {
                                                                                        Points =
Points * 5;// baixa

                                                                                } else {
                                                                                        Points =
Points * 7;// media

                                                                                }
                                                                        } else {
                                                                                if (TER == 2) {
                                                                                        if (TED < 20)
{

                                                                                                Points
= Points * 5;// baixa

                                                                                        } else {
                                                                                                if
(TED < 51) {

                                Points = Points * 7;// media

                                                                                                } else
{

                                Points = Points * 10;// alta
```

```
                                                                                    }
                                                                                }
                                                                            } else {
                                                                                if (TER == 3)
{
                                                                                    if
(TED < 20) {

        Points = Points * 7;// media
                                                                                    } else
{

        Points = Points * 10;// alta
                                                                                    }
                                                                                }
                                                                            }
                                                                        }

                                                                    } else {
                                                                            Points = 5;
                                                                    }


                                                                    PointsT = PointsT + Points;
                                                                    Points=1;
                                                                } catch (Exception E) {
                                                                    // Excecao para nao atributos
                                                                }

                                                            }

                        return PointsT;
                    }
}
```