Avaliação de Qualidade de Documentos de Requisitos para Engenharia de Software Orientado a Agentes

Emanuel Batista dos Santos[1]

[1]Centro de Informática – Universidade Federal de Pernambuco (UFPE)
Caixa Postal 7851 – 50732-970 – Recife – PE – Brazil

ebs@cin.ufpe.br

Abstract. *This paper presents an approach for evaluation of agent-oriented models in requirement phase. We make an overview of agent-oriented software development metodologies and software measurement concepts. This work is focused in metodologies baseds on models of i* [Yu 1995] framework like Tropos metodology [Castro et al. 2001]. And we show how define metrics for i* models using a framework [Franch 2006] for this.*

Resumo. *Esse artigo apresenta uma abordagem para avaliar documentos de requisitos de metodologias orientadas a agentes. São mostrados trabalhos relacionados a avaliação de documentos de requisitos e a obtenção de métricas para engenharia de requisitos. O trabalho é focado nas metodologias baseadas em i* [Yu 1995] como a metodologia Tropos [Castro et al. 2001]. E são mostradas formas de criar métricas [Franch 2006] para medir os modelos produzidos com i* na fase de requisitos.*

1. Introdução

As metodologias de desenvolvimento orientado a agentes são baseadas no conceito do agente. Elas tentam usar essa abstração na construção de software. Por ser uma iniciativa nova a orientação a agentes ainda não possue a mesma maturidade que a orientação a objetos. Para o desenvolvimento do software ainda faltam muitos elementos de processo e gerência nessas metodologias. Entre as atividades que ainda não são atendidas por essas metodologias está a medição de software.

Esse artigo apresenta uma motivação para a adoção de medidas de software para metodologias de desenvolvimento orientadas a agentes. Nele é apresentada um revisão da literatura mostrando como é feita a medição de software, especialmente na fase de requisitos. E apresenta uma forma de medir os modelos produzidos na fase de requisitos na metodologia Tropos.

O resto do artigo está estruturado da seguinte forma: a seção 2 apresenta conceitos básicos sobre a orientação a agentes e a importância da medição nessas metodologias. A seção 3 apresenta uma explanação de como a medição é vistas em diferentes modelos de qualidade e as principais métricas para engenharia de requisitos. A seção 4 mostra uma abordagem para medição na fase de requisitos de uma metodologia orientada a agentes. A seção 5 são feitas as considerações finais sobre esse artigo.

2. Desenvolvimento Orientado a Agentes

Com o desenvolvimento da rede mundial de computadores e de uma maior capacidade processamento das máquinas as aplicações também evoluíram. Elas passaram a ser mais complexas sendo capazes de manipular uma grande quantidade de informação. Além disso também passaram a ser distribuídas, autônomas e interligadas formando redes de dependências entre sistemas de software e usuários. Exemplos dessas aplicações são o comércio eletrônico, ensino à distância, aplicações multimídia entre outras [Garcia 2003].

Uma forma de conseguir atender as demandas dos software modernos é adotando um paradigma que se adeque a essa realidade. O desenvolvimento de software orientado a agentes é um desses paradigmas que pode ajudar a manipular uma complexidade maior. O software orientado a agente é o produto desse desenvolvimento. Segundo [Wooldridge 2002] software orientado a agentes pode ser definido como um software que atende as seguintes características:

Situado O software sente o ambiente e executa ações para alterar o ambiente;

Autônomo Tem controle sobre suas ações e estados internos, pode atuar sem intervenção direta de humanos;

Flexível Responde a alterações no ambiente, é orientado a objetivos/metas, oportunista, toma iniciativa;

Social Interage com outros agentes artificiais e humanos para completar suar tarefas ou ajudar os outros.

De acordo com essa definição o software orientado a agentes pode ser visto como uma forma de desenvolvimento que utiliza como abstração o agente. Segundo [Wooldridge 2002] um agente pode ser descrito como uma entidade ativa, autônoma, reativa, e com habilidade social. Uma outra definição de agente pode ser encontrada em [Ferber 1999], nessa definição um agente é uma entidade física ou virtual:

- Que é capaz de atuar em um ambiente
- Que pode comunicar diretamente com outros agentes
- Que é dirigida por um conjunto de tendências (na forma de objetivos individuais ou de uma função de satisfação/sobrevivência que precisa ser otimizada)
- Que possue recursos próprios
- Que é capaz de perceber seu ambiente (mas em uma extenção limitada)
- Que tem somente uma representação parcial do seu ambiente (e talvez nenhuma do todo)
- Que possue conhecimentos e pode oferecer serviços
- Que pode ser capaz de se reproduzir
- Cujo comportamento pretende satisfazer seus objetivos, levando em conta os recursos e conhecimentos disponíveis para isso e dependendo de suas percepções, representações e comunicação que recebe.

Muitas metodologias são usadas para o desenvolvimento orientado a agente, entre elas podemos citar Gaia [Wooldridge et al. 2000], Tropos [Castro et al. 2001][Bresciani et al. 2004], MaSE [Deloach et al. 2001] e Prometheus [Padgham and Winikoff 2002]. A maior parte dessas metodologias está preocupada com o desenvolvimento de software a partir de uma solução sem a preocupação com o entendimento do problema antes de construir o software. A figura 1 mostra a abrangência de algumas dessas metodologia.

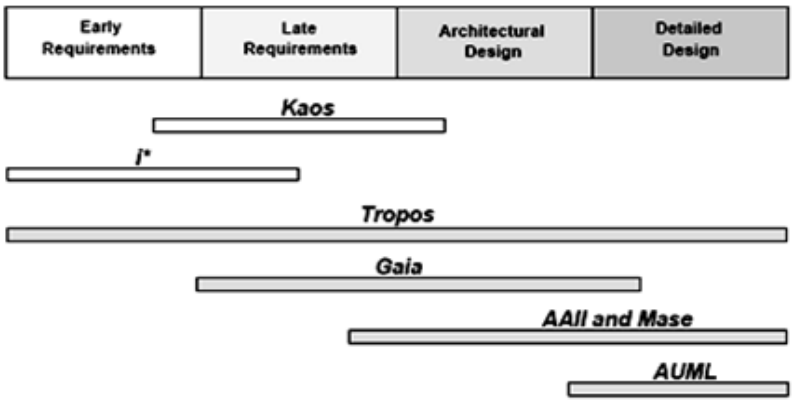

Figura 1. Metodologias de DOA [Bresciani et al. 2004]

Entre essas metodologias a que possui a melhor cobertura em termos de fases é a metodologia Tropos. Ela abrange desde as fases de requisitos até a implementação. As outras metodologias não se preocupam com a fase de requisitos deixando ao usuário a responsabilidade de gerir essa atividade. A partir desse ponto quando tratar de metodologias de desenvolvimento orientado a agentes o artigo irá focar na metodologia Tropos. A fase de requisitos é o principal diferencial dessa metodologia. Por isso a medição na fase de requisitos é uma atividade importante para a metodologia Tropos. A seção 3.3 mostra métricas para engenharia de requisitos. E a seção 4 mostra como pode ser feita a definição de métricas para modelos produzidos na fase de requisitos da metodologia Tropos.

A metodologia Tropos é baseada no framework i* [Yu 1995]. Esse framework permite definir modelos que representam os requisitos do sistema através de dependências entre atores estratégicos. O ator estratégico é uma forma genérica de representar os envolvidos nos processos do sistema. Eles tanto podem ser atores humanos, no caso os stakeholders, como atores de software que representam o sistema em si. A representação usando i* permite entender o "porquê" da construção do sistema além do "como", isso ocorre devido a captura de aspectos organizacionais e não-funcionais durante a modelagem. O framework i* e a metodologia Tropos serão detalhados a seguir.

2.1. Framework i*

O framework i* [Yu 1995] é um framework de modelagem conceitual desenvolvido para modelagem e análise. É usado para representar relacionamentos entre atores estratégicos em uma rede social. Isso é feito sob uma visão estratégica e intencional, de processos que envolvem vários participantes, ou seja, uma modelagem organizacional.

Para realizar esta análise, os engenheiros de requisitos modelam os stakeholders como atores e suas intenções como metas (goals). Os atores são entidades que possuem metas e podem depender de outros atores para conseguir alcançá-las. O conjunto de dependências criadas pelos atores formam uma rede social que representa o ambiente do sistema e o sistema em si.

O i* é formado por dois modelos: o modelo de Dependência Estratégica (SD, Strategic Dependency model) e o modelo de Razão Estratégica (SR, Strategic Rationale model). O modelo SD fornece uma descrição intencional de um processo em termos de uma rede de relacionamentos de dependência entre atores relevantes, ou seja, as relações

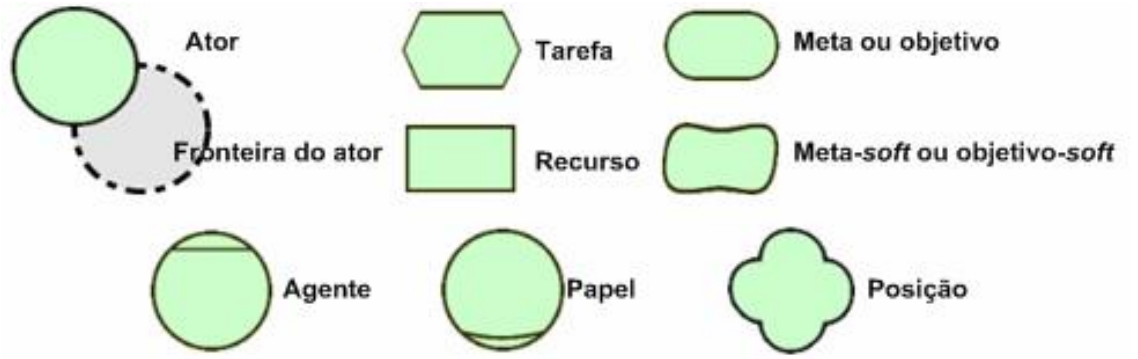

Figura 2. Elementos de Modelagem i*

de dependências externas entre os atores da organização. O modelo SR, por sua vez, apresenta uma descrição estratégica do processo, em termos de elementos do processo e das razões que estão por detrás deles. Fornece uma análise meio-fins de como as metas podem ser cumpridas através das contribuições dos demais atores. Os elementos de modelagem do i* estão na figura 2.

A estrutura conceitual do framework i* é utilizada para obter uma compreensão mais apurada sobre os relacionamentos da organização, de acordo com os diversos atores do sistema. Além disso o i* permite compreender as razões envolvidas nos processos de decisões. O i* possui diversas áreas de aplicação além da especificação de requisitos, tais como reengenharia de processos de negócio, desenvolvimento orientado a agentes análise de impactos organizacionais e modelagem de processos de software [Yu 1995].

2.2. Metodologia Tropos

Tropos é uma proposta de desenvolvimento de sistemas orientada a agentes, inspirada na análise de requisitos e fundamentada em conceitos sociais e intencionais [Castro et al. 2002]. Sua abordagem utiliza os modelos e conceitos oferecidos pelo framework i* [Yu 1995]. Tanto o modelo SD quanto o modelo SR do i* são usados por Tropos para capturar as intenções dos stakeholders, as responsabilidades do novo sistema em relação a estes stakeholders, a arquitetura do novo sistema e os detalhes de seu projeto.

As duas características fundamentais da metodologia Tropos são: (i) o uso de conceitos de nível de conhecimento, tais como agente, meta, plano e outros, durante todas as fases do desenvolvimento de software; e (ii) o importante papel atribuído à análise de requisitos quando o ambiente e o sistema a ser desenvolvido são analisados [Castro et al. 2002]. Além disso, Tropos propõe um framework de modelagem que visualiza o software sob cinco perspectivas complementares, [Bresciani et al. 2004]:

Social quais são os atores relevantes, o que eles querem? Quais são suas obrigações? Quais são suas capacidades?

Intencional quais são as metas relevantes e como elas se relacionam? Como elas estão sendo reconhecidas e quem solicita as dependências?

Comunicativa como os atores dialogam e como eles podem interagir uns com os outros?

Orientada a processos quais são os processos de negócio ou computação relevantes? Quem é responsável pelo que?

Orientada a objetos quais são os objetos e classes relevantes, bem como seus relacionamentos?

Tropos oferece um framework que engloba as principais fases de desenvolvimento de software, com o apoio das seguintes atividades: Requisitos Iniciais, Requisitos Finais, Projeto Arquitetural e Projeto Detalhado. Recentemente o escopo do Tropos foi estendido passando a incluir técnicas para geração de uma implementação a partir dos artefatos gerados na fase de Projeto Detalhado. Esta fase complementa propostas para plataformas de programação orientada a agentes. Tropos visa, entre outros objetivos, a definição de arquiteturas de software mais flexíveis, robustas e abertas. Além disso, estão sendo realizados esforços para tornar a metodologia mais compatível com o paradigma de programação orientada a agentes, bem como para dar um melhor suporte a áreas de aplicação baseadas na Web, tais como telecomunicações e comércio eletrônico. Contudo, Tropos ainda não contempla explicitamente algumas atividades de desenvolvimento de software tradicional, tais como testes, distribuição, gerência de configuração, entre outras.

Como é proposito desse trabalho é investigar a medição dos documentos de requisitos produzidos pela metodologia Tropos são detalhadas abaixo as fases de requisitos da metodologia.

2.2.1. Requisitos Iniciais

A fase de Requisitos Iniciais está preocupada com o entendimento de um problema estudando uma configuração organizacional existente [Silva 2003]. Durante esta fase, os engenheiros de requisitos modelam os stakeholders como atores e suas intenções como metas. Cada meta é analisada do ponto de vista de seu ator resultando em um conjunto de dependências entre pares de atores. As saídas desta fase são dois modelos:

1. O modelo de dependência estratégica que captura os atores relevantes, suas respectivas metas e suas interdependências; e
2. O modelo de razão estratégica que determina através de uma análise meio-fim como as metas podem ser cumpridas através das contribuições de outros atores.

A figura 3 mostra um exemplo de modelo SD na fase de requisitos inicias. Ele foi retirado de [Castro et al. 2002] e representa a modelagem de uma loja de mídia (MediaShop).

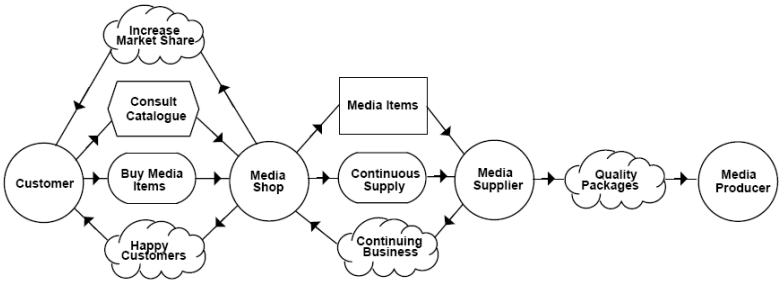

Figura 3. Exemplo de Modelo SD na fase de requisitos iniciais [Castro et al. 2002]

2.2.2. Requisitos Finais

A fase de Requisitos Finais introduz o sistema a ser desenvolvido como um outro ator no modelo de dependência estratégica [Silva 2003]. O ator que representa o sistema é

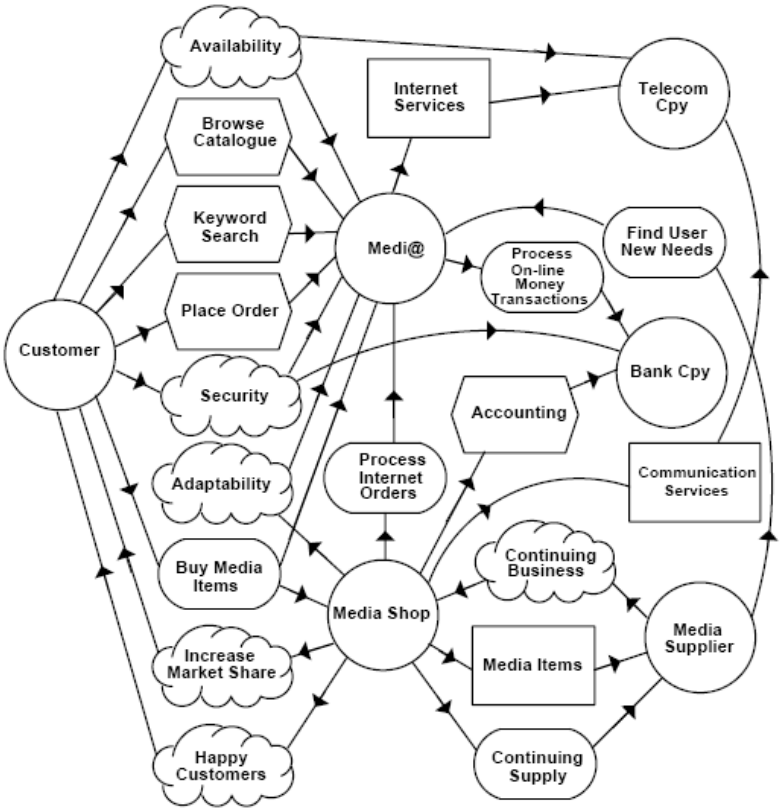

Figura 4. Modelo SD na fase de requisitos finais [Castro et al. 2002]

relacionado aos atores sociais em termos de dependências. Considera-se este ator que representa o sistema e se faz uma análise meio-fim para produzir um novo modelo de razão estratégica. Suas metas são analisadas e irão eventualmente levar a revisar e adicionar novas dependências com um subconjunto de atores sociais (os usuários). Se necessário decompõe-se o sistema que representa o ator em vários sub-atores e se revisa os modelos de razão e dependência estratégica.

A figura 4 mostra uma evolução da figura 3. Nela é adicionado o sistema e são feitos os relacionamentos entre o sistema e os atores criados no diagrama SD anterior.

A figura 5 mostra o SR do sistema da figura 4. Nessa fase é feita a expansão da fronteira do ator com o propósito de refinar os relacionamentos do ator para obter as razões que motivam os objetivos do sistema.

3. Medição de Software

A medição de software é uma prática que não é independente das demais e interfere com todos os subprocessos do desenvolvimento de software [Rifkin and Cox 1991]. Devido à má interpretação de conceitos de métricas e a dificuldades em coletar, armazenar, reportar e analisar os resultados, a implantação de um programa de métricas pode se algo custoso e controverso.

Segundo o IEEE [Committee 1992], a utilização de métricas reduz a subjetividade na avaliação da qualidade de software através do fortalecimento de informações quantitativas a respeito do produto e do processo, além de tornar a qualidade do software mais

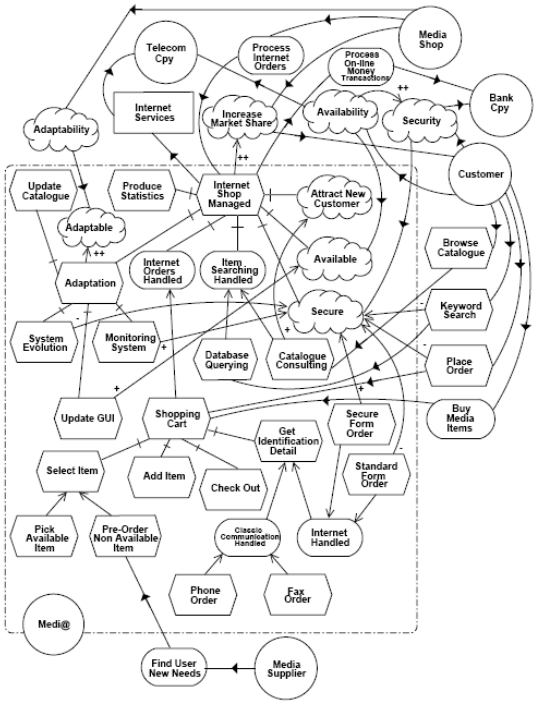

Figura 5. Modelo SR na fase de requisitos finais [Castro et al. 2002]

visível. A engenharia de software começa a evoluir para um caminho em que medições fazem parte dos processos básicos para se construir software. Informações quantitativas são utilizadas para orientar a tomada de decisões, e para validar ou refutar novas propostas de processos ou novas tecnologias, entre outros aspectos mensuráveis nas organizações de software. Trabalhando assim, as organizações passam a não serem guiadas apenas por opiniões, idéias ou suposições, e sim por dados objetivos e fatos devidamente embasados.

Segundo o PSM [McGarry et al. 2002], medições de software fornecem informações objetivas para suportar os gerentes de projetos principalmente nos seguintes aspectos:

- Comunicação efetiva através das informações objetivas;
- Acompanhamento dos objetivos dos projetos através da medição dos processos e produtos;
- Identificação e correção dos problemas de forma antecipada, uma vez que medições viabilizam uma estratégia de gerência pró-ativa;
- Suporte na tomada de decisões críticas;
- Justificativas para a tomada de decisões.

É unânime que os custos para a definição e implantação de um programa de métricas são elevados, e precisam ser respaldados por ganhos reais e visíveis para então serem apoiados pela alta gerência. A visão de Humphrey consolida de forma objetiva em [Humphrey 1989] as principais motivações para medição de software. Na Figura 6 estão apresentadas as principais motivações para a medição de software, segundo Humphrey:

Entender Métricas de software podem ser utilizadas para se adquirir aprendizado sobre algum aspecto do produto ou atividade do processo.

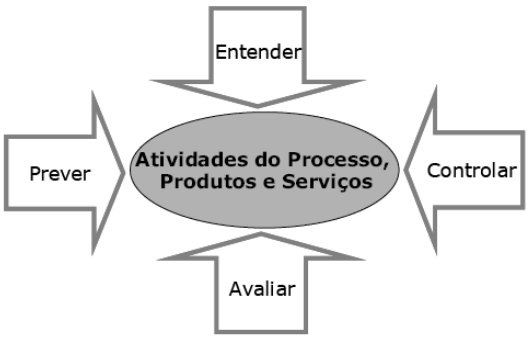

Figura 6. Principais Razões para medir qualidade segundo Humphrey

Avaliar Métricas de software podem ser utilizadas para se avaliar produtos e processos, verificando se os mesmos atendem a seus critérios de aceitação.

Controlar Dados podem ser utilizados para controlar o desenvolvimento de software, assim como em diversas outras áreas como engenharia e manufatura, por exemplo.

Prever Métricas de software podem ser utilizadas como base para a elaboração de estimativas, na construção de médias e tendências.

3.1. Medição em Modelos de Qualidade

Nessa seção serão descritos, de forma sucinta, alguns modelos que definem orientações e processos macro para a implantação de programas de medições: o PSM - Practical Software Measurement [PSM 2007], a Norma ISO/IEC 15939 [ISO/IEC 2002] e o CMMI [Chrissis et al. 2003]. Além dos modelos de processo, iremos descrever também o Paradigma GQM, proposto por Basili [Basili et al. 1994], que apesar de não ser um modelo, é uma técnica que provê suporte a todos os modelos propostos pela literatura de métricas e aborda o aspecto de definição de métricas a partir de objetivos estratégicos.

3.1.1. PSM

O PSM é um processo de medição orientado a informações que endereça os objetivos técnicos e gerenciais de uma organização, e suas orientações refletem as melhores práticas utilizadas por profissionais das áreas de engenharia de software e de sistemas. Surgiu como uma iniciativa do Departamento de Defesa norte-americano em 1994 e foi publicado pela primeira vez em 1997, sob a forma de um manual. Atualmente já sem encontra publicado em um livro, Practical Software Measurement [McGarry et al. 2002].

O PSM busca prover informações objetivas aos gerentes de projetos, necessárias para o atendimento dos custos, cronogramas e objetivos técnicos dos projetos. É um processo flexível e baseado nas melhores práticas de medição do Departamento de Defesa norte-americano, sendo também compatível com a norma ISO/IEC 15939.

3.1.2. A Norma ISO/IEC 15939

Este padrão internacional ISO/IEC propõe um processo de medição aplicável a organizações de software e a organizações que possuem relacionamento com a enge-

nharia de software [ISO/IEC 2002]. Possui como propósito a definição das atividades e tarefas que são necessárias para a implantação de programas de medições. A norma provê também definições para os principais conceitos relacionados a medições que são comumente utilizados na indústria de software. Esse padrão não tem o objetivo de definir métricas de software para serem aplicadas aos projetos das organizações, ele busca fornecer orientações de processo para suportar a identificação das medições que enderecem os objetivos exclusivos dos projetos e organizações.

O campo de aplicação desta norma é bastante abrangente, vai desde empresas que fornecem software a empresas que adquirem software. Pode ser utilizada pelos fornecedores para avaliar seus processos de desenvolvimento e também a qualidade de seus produtos fornecidos, e também pelas organizações que adquirem software, que podem utilizar o padrão para suportar o gerenciamento do contrato da aquisição e para avaliar a qualidade do produto adquirido [ISO/IEC 2002].

3.1.3. Medição e Análise no CMMI

O modelo CMMI criou a área de medição e análise visando prover ao mercado as orientações a respeito de como se implantar medição e análise. A área chave tem como objetivo desenvolver e manter a capacidade de medição em uma organização ou projeto para prover as necessidades de informações da gerência na tomada de decisões [Zubrow 2001]. Por ter sido definida a partir dos conceitos introduzidos pelo PSM e pela norma ISO 15939, a área de processo consolida as melhores práticas e acrescenta melhorias extraídas a partir das experiências prévias do PSM e da própria norma ISO. Nesse contexto, o caminho mais natural é que as organizações passem a adotar o CMMI, ao invés dos demais modelos, visto que o mesmo fornece orientações de processo para outras áreas e é um modelo mais atual que surgiu para substituir o CMM.

3.1.4. O Método GQM

O paradigma GQM (Goal Question Metric Paradigm) foi criado por Basili com o objetivo de suportar as organizações na institucionalização de processos de medições, especificamente na identificação de objetivos que serão traduzidos em medições quantitativas [Basili et al. 1994]. O GQM define orientações para:

- Definir os principais objetivos que serão tratados no programa de medições (Goal);
- Identificar um conjunto de questões que ajudem o atendimento dos objetivos (Question);
- Definição e recuperação de dados que respondam as questões identificadas (Metrics).

O modelo GQM possui uma estrutura hierárquica composta por três níveis, conforme ilustrado na Figura 7 . O nível 1 é considerado o nível conceitual, o nível dos objetivos. Objetivos são estabelecidos para as diversas entidades de uma organização, como por exemplo: seus projetos, produtos e processos. O nível 2 é considerado o nível operacional, o nível das questões. As questões são identificadas para caracterizar o alcance de um objetivo específico. O nível 3 é considerado o nível quantitativo, o nível das métricas. Um

conjunto de dados é associado a uma determinada questão com o objetivo de respondê-la de maneira quantitativa.

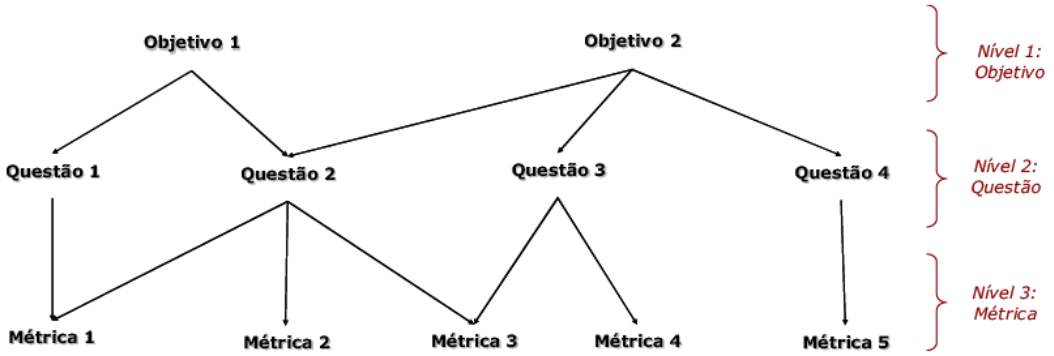

Figura 7. Paradigma GQM (Goal Question Metric)

O modelo GQM compõe uma estrutura hierárquica, conforme ilustrado na Figura 7, iniciando com um objetivo que deve possuir o propósito da medição, a entidade a ser medida (produto, processo ou recursos), o atributo a ser medido e o ponto de vista a partir do qual a medição deverá ser extraída.

O objetivo é refinado em diversas questões. Por sua vez, cada questão é refinada em métricas, podendo ser elas objetivas ou subjetivas. Questões podem ser identificadas para mais de um objetivo, e métricas também podem pertencer a mais de uma questão e conseqüentemente endereçar vários objetivos. As métricas devem endereçar os diferentes pontos de vista de cada objetivo.

3.2. Métricas de Software

Segundo Norman Fenton [Fenton and Pfleeger 1996], medição é o processo através do quais símbolos e números são atribuídos a atributos de entidades do mundo real de uma maneira que os mesmos possam ser descritos através de regras claramente definidas. Uma entidade é uma pessoa, lugar, evento, período de tempo ou um outro objeto que será caracterizado pela medição [ISO/IEC 2002]. Um atributo é uma característica ou propriedade de uma entidade [McGarry et al. 2002], [ISO/IEC 2002]. Por fim precisamos definir o sistema de mapeamento, que são as regras que definem como os atributos serão medidos.

Em [Feitosa 2004] é possível encontrar um levantamento dos principais conceitos sobre métricas, eles estão resumidos a seguir.

Métrica básica É a medição de um único atributo de uma entidade através de um sistema de mapeamento. A mesma é independente de outras medidas e captura informação a respeito de um único atributo.

Método de medição Seqüência lógica de operações utilizadas na quantificação de um determinado atributo em relação a um determinado valor.

Escala de medição É um conjunto ordenado de valores, contínuo ou discreto, ou um conjunto de categorias, às quais os atributos são mapeados.

Unidade de medida Uma unidade de medida é uma quantidade definida ou adotada por convenção, através da qual outras quantidades do mesmo tipo podem ser comparadas com o objetivo de expressar sua magnitude em relação à quantidade sendo mensurada.

Métrica Derivada É a medição definida como função de duas ou mais medições básicas ou derivadas. Segundo o CMMI, a função para caracterizar uma medição derivada deve ser uma função matemática entre duas ou mais medições básicas.

Indicador Indicadores são métricas que fornecem informações adicionais de estimativas ou avaliações de um determinado atributo . Eles representam a base para as atividades de análise e para a tomada de decisões. São definidos à partir de um modelo de análise que define a relação entre duas ou mais medições básicas e derivadas. Um indicador compara a métrica com um resultado esperado. Os indicadores permitem a tomada de decisões através da visão da real situação dos aspectos de um projeto.

A medição é uma das formas de evitar erros. Ela permite ter informações objetivas sobre o comportamento do objeto medido. Na engenharia de software ela é feita através de métricas. As métricas podem ser categorizadas em métricas de processos, produtos e requisitos [Ali 2006].

As métricas de produtos avaliam o produto de software no estado de desenvolvimento. Elas podem ser aplicadas da fase de requisitos até a fase de instalação. Elas podem medir tamanho de software, numero de páginas de documentação e complexidade de projeto. Esses tipos de métricas são largamente utilizadas na industria por serem mais conhecidas e faceis de aplicar. Um exemplo de métrica de produto é a contagem de linhas de código (LOC), ela ter um estimativa simples de quanto software foi feito.

As métricas de processo medem o processo de desenvolvimento. Elas incluem tipos de metodologias usadas, nível de experiência dos recursos humanos e todo tempo de desenvolvimento. O objetivo dessas métricas é facilitar a definição de atividades e agenda durante o processo de software. Quando aplicadas as métricas de processo esperasse que elas informem sobre a produtividade, detecção de erros e eficiência dos processos de forma objetiva.

As métricas de requisitos são a outra forma de medir o software nas fases iniciais, elas serão detalhadas na subseção 3.3.

3.3. Métricas para Engenharia de Requisitos

Um dos objetivos da engenharia de requisitos é procurar concordância entre o que é produzido e o que é esperado, para que o acordo entre todas as partes afetadas seja cumprido. Já a algum tempo trabalhos experimentais [Costello and Liu 1995] comprovam que a fase de requisitos é a melhor etapa para identificar erros e inconcistências no software. Isso ocorre por ser mais barato concertar erros nessa fase e evitar que eles se propagem para as posteriores.

No caso da engenharia de requisitos existem métricas já conhecidas que podem ser categorizadas [Costello and Liu 1995, Ali 2006] em: métricas de tamanho, rastreabilidade de requisitos, volatilidade de requisitos e completude de requisitos.

As métricas de tamanho ou de Use Case são métricas que medem o tamanho do documento de requisitos. Elas usam o modelo de casos de uso como forma de medida. Podem ser medidos número de casos de uso, o número de funções cobertas, ou número de casos uso analisados e número de fluxos alternativos.

As métricas de rastreabilidade provem informações para determinar se todos os

relacionamentos e dependências estão sendo endereçados. Essas métricas ajudam a reduzir interpretações erradas de outras métricas. Também leva a um baixo nível de requisitos com origens inválidas. As principais métricas de rastreabilidade são o número de níveis de dependências entre requisitos relacionados, a quantidade de requisitos com ligações quebradas, o número de requisitos sem ligações, o número de ligações inconsistentes acima e abaixo do nível do requisito, entre outras métricas.

As métricas de completude são usadas para verificar se os requisitos estão completos. Como completo pode se entender que ele tem todas as páginas numeradas, todas as figuras e tabelas com legendas e numeradas, todas as referências presentes e todas as seções e subseções numeradas. Essas métricas permitem verificar a estrutura do documento de requisitos e ver se todos os requisitos estão devidamente especificados, ou se faltam requisitos ainda não especificados no nível correto.

As métricas de volatilidade são métricas sobre o período de tempo que os requisitos são alterados. Essas métricas são usadas para verificar as razões sobre as alterações dos requistos. Elas verificam o tempo entre adição e remoção, entre atualizações sucessivas e o número de vezes que isso ocorre. É esperado que haja uma maior volatilidade no início do projeto mas que ela diminua com o tempo. Projeto onde a volatilidade dos requisitos é muito alta no final do projeto podem indicar algum problema no projeto.

A maior parte das métricas para requisitos apresentadas não se adequam a medição de modelos baseados em agentes devido a características particulares dessa abordagem. Assim é necessário criar métricas que permitam medir a qualidade desse modelos de forma coerente. Uma abordagem para se fazer isso é mostrada na seção 4.

4. Métricas para modelos baseados em agentes

A avaliação de qualidade é algo complexo até mesmo para especialistas, utilizar métricas permitiria fazer isso de uma forma sistemática. A necessidade de avaliar a qualidade dos modelos produzidos na engenharia de requisitos como o framework i*.

Para medir modelos i* existem poucas abordagens, a abordagem adotada será a proposta por Xavier Franch [Franch et al. 2004, Franch 2006] . Essa abordagem descreve um framework para definição de métricas baseado na avaliação estrutural dos modelos i*. Para definir uma métrica é necessário definir o que medir. Por isso, o primeiro passo é definir as propriedades que serão avaliadas. Várias propriedades podem ser medidas como a corretude, segurança, modularidade entre outros. Depois de definida a propriedade que se quer medir o próximo passo é definir os indicadores de medição.

Há duas formas de definir através do tipo de retorno ou do escopo da medição. Quanto ao tipo de retorno os indicadores retornam:

Indicadores numéricos retornam valores que varia em um faixa numérica
Indicadores lógicos indicam se a propriedade foi atendida ou não
Indicadores de modelo-elemento retornam um conjunto de elementos que atende a uma propriedade

Quanto ao escopo da medição os indicadores podem ser globais quando produzem um único valor para qualquer tipo de elemento, locais quando produzem um valor para cada tipo específico de elemento e em grupo quando produzem valores para várias combinações de valores de acordo com um critério de agrupamento.

	Dependency		Attribute		
f_D	Sort	Type	DP	DA	PA
	G	any	0	0	0
	SG	any	0	0	0
	T	any	0	0	0
	R	H-H	1	0,6	1
		H-S	0,9	0,8	0,7
		S-S	0,8	1	0,5
h_A	Sort		DP	DA	PA
	H		0,7	0,7	0,8
	S		1	1	1

Figura 8. Tabela das funções para definição de métricas

Na abordagem de [Franch 2006] a medição é feita através da atribuição de pesos para os elementos estruturais do i* de acordo com a propriedade que se quer medir. Os elementos estruturais são as dependências e os atores que compõe o modelo. Esses elementos são contados e são utilizadas funções para atribuir valor ao modelo de acordo com o resultado da contagem e o peso atribuído a cada elemento. Como resultado dessa métrica está um score que vai representa quanto o modelo avaliado para uma propriedade. Esse score permite comparar modelos diferentes de acordo com uma propriedade. Por exemplo, se fossem feitas diferentes modelagens para sistemas web utilizando i* seria possível comparar esses sistemas de acordo com critérios de qualidade de uma forma sistemática.

	Current system	Proposed system with human meeting scheduler	Proposed system with software meeting scheduler
DP	1	0,64	0,55
DA	0,42	0,63	1
PA	1	0,66	0,74
RD	0,64	1	1

Figura 9. Tabela com os resultados das medições

A figura 8 é um exemplo da tabela de métricas criadas para avaliar atributos de qualidade em uma solução. Nela é possível ver os pesos diferentes atribuídos aos atributos em duas funções h_A e f_D que servem respectivamente para medir os atores e dependências dos modelos i*.

A figura 9 mostra o resultado das métricas definidas com base nos pesos atribuídos como mostra a figura 8. Nela é possível ver que cada atributo possui um score relacionado com o modelo que está sendo avaliado.

Outra abordagem de medição é o AGORA [Kaiya et al. 2002] uma metodologia de avaliação de qualidade baseada em objetivos que avalia a qualidade da especificação de requisitos em função da corretude, não ambiguidade, completude, consistência, verificabilidade, modificabilidade, rastreabilidade, grau de importância e estabilidade. Essa

metodologia avalia elementos da decomposição de objetivos de acordo com os critérios prioridade dos stakeholders. A principal diferença dessa abordagem para a outra é que o AGORA atua sobre as decomposições e o framework de Xavier Franch atua sobre a estrutura dos modelos.

5. Conclusões

A medição se mostra como uma necessidade para melhorar a qualidade dos artefatos de software produzidos. Para as metodologias orientadas a agente isso ainda constitui um desafio por não haver modelos de medição bem definidos. Esse trabalho mostrou uma breve revisão da literatura sobre medição e métricas para engenharia de requisitos. E introduziu uma motivação para a medição de modelos i*. Mostrando um framework [Franch 2006] que será utilizado na atividade de definição de métricas para medir documentos orientados agentes.

5.1. Trabalhos Futuros

Esse é um trabalho inicial e ainda depende de outras atividade para consolidar as idéias aqui apresentadas. Entre os trabalhos futuros é possível enumerar:

1. Definição de critérios para medição de modelos i*
2. A escolha de propriedades que serão medidas
3. Definição das métricas para avaliação
4. Aplicação dessas métricas em um estudo de caso

Referências

Ali, M. J. (2006). Metrics for requirements engineering. Master's thesis, Umea University, Department of Computing Science, Sweden.

Basili, V. R., Caldiera, G., and Rombach, H. D. (1994). The goal question metric approach. *Encyclopedia of Software Engineering*, pages 528–532.

Bresciani, P., Perini, A., Giogini, P., Giunchiglia, F., and Mylopoulos, J. (2004). Tropos: An agent oriented software development. In *Proceedings of Autonomous Agents and Multi-Agent Sytems*, pages 203–236, Netherlands. Kluwer Academic Publishers.

Castro, J., Kolp, M., and Mylopoulos, J. (2001). A requirements-driven development methodology. In *13th International Conference on Advanced Information Systems Engineering (CAiSE'01)*, volume LNCS Vol. 2068, pages 108–123, Interlaken, Switzerland. Springer-Verlag.

Castro, J., Kolp, M., and Mylopoulos, J. (2002). Towards requirements-driven information systems engineering: the tropos project. *Information Systems*, 27(6):365–389.

Chrissis, M. B., Konrad, M., and Shrum, S. (2003). *CMMI Guidelines for Process Integration and Improvement*. Addison Wesley.

Committee, I. S. (1992). Standard for a software metrics methodology. Technical report, IEEE Software.

Costello, R. J. and Liu, D.-B. (1995). Metrics for requirements engineering. *J. Syst. Softw.*, 29(1):39–63.

Deloach, S. A., Wood, M. F., and Sparkman, C. H. (2001). Multagent systems engineering. *International Journal of Software Engineering and Knowledge Engineering*, 11(3).

Feitosa, C. M. A. C. (2004). Definição de um processo de medição e análise com base nos requisitos do cmmi. Master's thesis, Centro de Informática, Universidade Federal de Pernambuco.

Fenton, N. E. and Pfleeger, S. (1996). *Software Metrics, A Rigorous Approach*. Int. Thomson Computer Press.

Ferber, J. (1999). *Multi-Agent System*. Addison Wesley Longman, Harlow.

Franch, X. (2006). On the quantitative analysis of agent-oriented models. In *Proceedings 18th International Conference in Advanced Information Systems Engineering, CAiSE*, pages 495–509.

Franch, X., Grau, G., and Quer, C. (2004). A framework for the definition of metrics for actor-dependency models. In *Requirements Engineering. 12th IEEE International Conference RE 2004*, Kyoto, Japan. IEEE Computer Society.

Garcia, A., L. C. Z. F. O. A. C. J., editor (2003). *Software Engineering for Large-Scale Multi-Agent Systems: Research Issues and Practical Applications*, volume LNCS Vol. 2603. Lecture Notes in Computer Science.

Humphrey, W. S. (1989). *Managing the Software Process*. Addison Wesley.

ISO/IEC (2002). *International Standard ISO/IEC FDIS 15939, Software Engineering - Software Measurement Process*. ISO/IEC.

Kaiya, H., Horai, H., and Saeki, M. (2002). Agora: Attributed goal-oriented requirements analysis method. In *RE*, pages 13–22.

McGarry, J., Card, D., Jones, C., Layman, B., Clark, E., Dean, J., and Hall, F. (2002). *Practical Software Measurement: Objective Information for Decision Makers*. Addison Wesley.

Padgham, L. and Winikoff, M. (2002). Prometheus: A pragmatic methodology for engineering intelligent agents. In *Proceedings of the workshop on Agent-oriented methodologies at OOPSLA*, Seattle.

PSM (2007). Psm: Practical software measurement. www.psmcs.com, Último acesso em 22/03/2007.

Rifkin, S. and Cox, C. (1991). Measurement in practice. Technical Report CMU/SEI-91-TR-016, Software Engineering Institute.

Silva, C. T. L. L. (2003). Detalhando o projeto arquitetural no desenvolvimento de software orientado a agentes: O caso tropos. Master's thesis, Centro de Informática, Universidade Federal de Pernambuco.

Wooldridge, M. (2002). *An Introduction to Multiagent Systems*. John Wiley and Sons.

Wooldridge, M., Jennings, N. R., and Kinny, D. (2000). The gaia methodology for agent-oriented analysis and design. *Autonomous Agents and Multi-Agent Systems*, 3(3):285–312.

Yu, E. (1995). *Modelling Strategic Relationships for Business Process Reengineering*. PhD thesis, Dept. of Computer Science, University of Toronto.

Zubrow, D. (2001). The measurement and analysis process area in cmmi. Technical report, Software Engineering Institute.