Elicitação da dificuldade de entendimento com base na teoria de Bergson

FRANCISCO SUPINO MARCONDES
Instituto Tecnológico de Aeronáutica (ITA) – São Paulo – Brasil
fsmarcondes@acm.org

DENIS ÁVILA MONTINI
Instituto Tecnológico de Aeronáutica (ITA) – São Paulo – Brasil
denisavila.montini@tcs.com

ÍTALO SANTIAGO VEGA
Pontifica Universidade Católica de São Paulo (PUC-SP) – São Paulo – Brasil
italo@pucsp.br

LUIZ ALBERTO VIEIRA DIAS
Instituto Tecnológico de Aeronáutica (ITA) – São Paulo – Brasil
vdias@ita.br

Resumo – Este estudo busca elicitar o problema de entendimento voltado à construção de software a fim de ser possível estudar formas de removê-lo apropriadamente. Para tanto faz uso da teoria da intuição de Bergson para investigar alguns aspectos relevantes deste problema e sua implicação na computação e no desenvolvimento de sistemas de software.

Palavras-chave: Requisitos de software, Projeto de software, Teoria da intuição, .

Introdução

Quando se vai construir um sistema de software, dificilmente é possível obter um perfeito entendimento entre o cliente e a equipe técnica, bem como entre os elementos da própria equipe sobre aquilo que deve ser construído. Por conta disso, chegou-se a conclusão de que para se construir um sistema de software é preciso continuamente verificar junto ao usuário se o entendimento sobre o problema está correto, tal conhecimento resultou numa das boas práticas de Booch [3] denominada *desenvolver software iterativamente*.

Outro aspecto importante é verificar se a equipe técnica está produzindo o produto correto, ou seja, continuamente deve-se confrontar os requisitos com o produto que está sendo construído, fato que resultou em outra boa prática denominada *verificar continuamente a qualidade do software* [3].

Ao analisar essas duas boas práticas, se verifica que ambas possuem a mesma natureza que é de perseguir, continuamente, o apropriado entendimento sobre especificação de requisitos visando confirmar a sua correção (*desenvolver software iterativamente*) e que o produto atenda a especificação de requisitos (*verificar continuamente a qualidade do software*), possuindo apenas uma diferença de grau entre elas.

Em outras palavras, pode-se dizer que, até o produto receber o aceite do cliente, em nenhum momento do desenvolvimento a equipe tem certeza de estar desenvolvendo o produto correto. Considerando que os projetos mal sucedidos devido a especificação, em geral, tem um conjunto de funcionalidades esperadas pelo cliente, o que implica em

que mesmo sem ter certeza de estar desenvolvendo o produto correto algumas funcionalidades estarão, mesmo que parcialmente, corretas; ou seja, a equipe técnica tem dificuldade para entender o problema do cliente, conseguindo apenas compreender partes ou instâncias dele. Este problema será denominado de entendimento relativo.

Este estudo busca elicitar o problema do entendimento relativo voltado à construção de software a fim de ser possível estudar formas de removê-lo propiciando assim melhor entendimento entre o cliente e entre os membros da equipe técnica e com isso diminuindo a necessidade de verificar continuamente a qualidade do software e do desenvolvimento iterativo e incremental; além de reduzir o risco do projeto uma vez que se tem certeza de se estar produzindo corretamente o produto correto.

Este estudo faz uso da teoria da intuição proposta por Bergson na década de 1930 e busca, apoiando-se nesta teoria, investigar alguns aspectos relevantes da implicação desta teoria ao desenvolvimento de sistemas de software. Como se trata apenas de uma investigação com intuito de elicitação do problema, não será apresentado nenhum resultado prático ou estudo de caso, mas sim uma discussão teórica sobre o tema.

Metodologia

Para este estudo foram utilizadas as estratégias de levantamento e de análise bibliográfica resultando numa investigação exploratória dos conceitos envolvidos.

Revisão Bibliográfica

Já é bastante conhecida a constatação do problema relacionado ao grande número de projetos de software que falham ou atrasam. Schach [5] mostra que 18% dos projetos são cancelados; 53% são finalizados com atraso, acima do orçamento e/ou com recursos a menos. Apenas 29% dos projetos são concluídos com sucesso. O Chaos Report [6], em 1999, contém o relato que em projetos com valores menores que $750K com 6 pessoas durante 6 meses há uma taxa de sucesso de 55%, enquanto num projeto de $3M a $6M com 18 pessoas e 18 meses a taxa cai para 15% e projetos de mais de $10M com mais de 500 pessoas e por mais de 36 meses a taxa de sucesso é de 0%.

Devido a tais problemas ao longo da história da Engenharia de Software vem se pesquisando formas de administrar ou resolver esses problemas, sendo que uma das formas que se tem utilizado são as boas práticas de Booch [3].

Kruchten [3] apud Grady Booch, buscando resolver diversos problemas comuns no desenvolvimento de sistemas de software recolheu com base em dados de projetos comerciais aquilo que ele chamou de melhores práticas de software. São elas*: 1) Desenvolver software iterativamente; 2) Gerenciar requisitos; 3) Usar arquiteturas baseadas em componentes; 4) Modelar visualmente o software; 5) Verificar continuamente a qualidade do software; e 6) Controlar mudanças do software*. Essas boas práticas vem permeando grande parte dos projetos de software contemporâneos. Neste estudo são feitas referências a desenvolver software iterativamente e verificar continuamente a qualidade do software.

A boa prática *desenvolver software iterativamente*, indica que o processo sequencial adia o risco do projeto aumentando o custo de corrigir erros de fases anteriores. Essa boa prática faz referência ao modelo Espiral de Boehm [4] e busca por meio de iteração e refinamentos sucessivos atacar ativamente os riscos identificando-os o mais cedo possível durante o desenvolvimento.

A boa prática *verificar continuamente a qualidade do software* afirma que, uma vez que quanto mais tarde se descobrem os problemas de software mais cara se torna repará-los, é importante avaliar continuamente se os aspectos funcionais, confiabilidade e desempenho estão de acordo com a especificação [3], ou seja, essa boa prática busca

atacar ativamente os riscos envolvidos na construção do projeto.

Bergson [1, 2], trabalha dois aspectos importantes da ciência tratando, entre outras coisas, da diferença do conhecimento obtido por meio de análises e do conhecimento obtido por meio da apreensão do problema, formando o que foi chamado de Teoria da Intuição. Tal Teoria pode ser definida “tanto como uma faculdade quanto como um modo de conhecimento distinto do intelectual, em que não caberia a interpretação metodológica” [7]. Segundo a Teoria, o entendimento obtido por meio de análises é relativo pois varia conforme a posição do observador enquanto o entendimento por meio da apreensão é absoluto pois é como se o pesquisador “entrasse” no problema conhecendo seu todo.

Discussão

O problema do entendimento relativo ocorre não apenas no contexto da Computação mas representa um problema puramente científico sendo abordado por diversos ramos da filosofia. Bergson [2] afirma que o entendimento de um problema se dá apenas por meio do absoluto que representa “entrar” em um problema, enquanto os métodos de análise por estudar o problema “de fora” possuem apenas uma visão relativa e parcial do problema.

Ao perceber o absoluto, a pessoa consegue compreender a verdadeira natureza do problema incluindo o seu ponto indescritível e, por isso, embora seja possível compreender o absoluto não é possível descrevê-lo. Para um melhor entendimento sobre este ponto, imagine o conjunto dos números reais. É possível compreender a sua totalidade, mas não existe uma forma de descrevê-lo com a mesma perfeição do entendimento. O mesmo princípio pode ser aplicado ao descrever um objeto de arte. Por mais descrições que sejam feitas, não é possível obter uma idéia absoluta como se a pessoa estivesse em contato direto com ela.

Logo, entre o absoluto e a sua descrição, infinitos fatos são omitidos deixando infinitas lacunas na descrição. Suponha a narrativa de um fato. Apenas os aspectos mais importantes são descritos deixando infinitos detalhes como roupas, temperatura, cores, árvores, eventuais animais, etc de fora da descrição por não serem relevantes, sendo por isso uma discretização da experiência absoluta e por isso relativa.

O problema é que as lacunas da descrição por mais completas que sejam são infinitas, seria o mesmo que tentar reproduzir uma cidade por meio de fotos. Por mais que se tire fotos não será possível uma visão do absoluto. No entanto a mente humana busca sempre buscar o absoluto fechando todas as lacunas e as que não são descritas são preenchidas pela mente por meio da imaginação.

Considere o caso da leitura de um romance, muitos detalhes como tipos das roupas, cômodos, expressões e trejeitos de personagens em geral não são descritos sendo deixados a cargo da imaginação. Se tratando de um romance isso não é um problema, mas se tratando de software pode ser, principalmente quando a imaginação da equipe difere da necessidade cliente, o que em geral acontece; levando à boa prática *desenvolver software iterativamente.*

Considere o caso de um interprete de música clássica, em geral, a criatividade do interprete surge nos pontos em que o autor não especificou criando interpretações bastante diversas entre um intérprete e outro. Imagine uma mesma música em que o andamento não tenha sido descrito pelo compositor. Um intérprete pode utilizar o andamento *presto* (veloz e animado) enquanto outro pode achar mais interessante o andamento *grave* (muito vagaroso e solene) e o resultado pode ser completamente diferente.

Quando aplicado a software, o mesmo espaço para imaginação surge nas lacunas da especificação de requisitos, nas quais o desenvolvedor pode escolher infinitas opções de preenchimento. Tal problema conduz à boa prática *verificar continuamente a qualidade*

do software.

No entanto em nenhum dos dois casos as boas práticas atacam o problema do entendimento relativo mas sim os sintomas por ele causados, de maneira que essas boas práticas buscam mascarar o problema do entendimento relativo e não resolvê-lo.

Por outro lado se há o conhecimento do absoluto, não há necessidade do uso da imaginação pois simplesmente se conhece o todo. Considere o caso de uma auto-biografia, ao discretizar o absoluto numa descrição, o autor, conforme o exposto cria infinitas lacunas que deverão ser preenchidas pela imaginação do leitor. Por outro lado, o autor, ao ler a sua auto-biografia, não recorre à imaginação e sim à lembrança do evento, sendo capaz de descrever quais eram realmente as roupas, cores, temperatura, etc sem recorrer ao que poderia ser.

Esta propriedade do conhecimento absoluto é o que difere o conhecimento real (absoluto) do fantasioso (relativo), ou seja, da verdadeira natureza do problema contra aquilo que a imaginação busca preencher. Quando aplicado este pensamento ao desenvolvimento de software se pode ver uma possível causa do problema do entendimento relativo tanto na análise de requisitos quanto no próprio projeto do software.

Uma vez que o não entendimento absoluto do problema pela equipe técnica traz entendimentos diferentes do problema. Isso dificulta tanto o entendimento com o cliente quanto o entendimento entre a equipe sobre aquilo que deve ser construído é dificultado, remetendo ao problema inicial apresentando no primeiro parágrafo deste estudo.

Conclusão

Este estudo buscou elicitar o problema do entendimento relativo voltado à construção de software a fim de ser possível estudar formas de removê-lo propiciando assim melhor entendimento entre o cliente e entre os membros da equipe técnica e com isso diminuindo a necessidade de verificar continuamente a qualidade do software e do desenvolvimento iterativo e incremental; e reduzir o risco do projeto uma vez que se tem certeza de se estar produzindo corretamente o produto correto.

Este estudo fez uso da teoria da intuição proposta por Bergson na década de 1930 e buscou, apoiando-se nessa Teoria, investigar alguns aspectos relevantes da implicação desta teoria ao desenvolvimento de sistemas de software. Como se trata apenas de uma investigação com intuito de elicitação do problema, não foi apresentado nenhum resultado prático ou estudo de caso, mas sim uma discussão teórica sobre o tema.

Para este estudo foram utilizadas as estratégias de levantamento e de análise bibliográfica resultando numa investigação exploratória dos conceitos envolvidos.

O estudo apresentou a diferença entre o entendimento real (absoluto) e o fantasioso (relativo) aplicado ao desenvolvimento de software tendo como objetivo lançar um análise sob um novo prisma ao problema do entendimento dos requisitos tanto do ponto de vista do analista como da equipe de desenvolvimento.

Referências

[1] Deleuze, G.; Bergsnonismo; Ed. 34, 1a Edição, São Paulo, 2008

[2] Bergson, H.; Revue de Métaphysique et de Morale; 1903

[3] Kruchten, P.; Introdução ao RUP; Ed. Ciencia Moderna, 2a Edição; Rio de Janeiro, 2004

[4] Boehm, B. W; Software Engineering Economics; Prentice Hall; Nova Jersey, 1981

[5] Schach, S. R.; Engenharia de Software: Os Paradigmas Clássico & Orientado a Objetos; McGrawHill, 7a Edição; São Paulo, 2009.

[6] Chaos Report, 1999

[7] Coelho, J. G; Bergson: Intuição e Método Intuitivo; Scielo: http://www.scielo.br/pdf/trans/v21-22n1/v22n1a12.pdf, visualisado em 07/2009.

Contato
Francisco Supino Marcondes - Curriculum Lattes: http://lattes.cnpq.br/4363386591250890
Denis Avila Montini - Curriculum Lattes: http://lattes.cnpq.br/6682778113989191
Ítalo Santiago veja - Curriculum Lattes: http://lattes.cnpq.br/1696460650494488
Luiz Alberto Vieira Dias - Curriculum Lattes: http://lattes.cnpq.br/8393302442560429