**ALGORITMOS GENÉTICOS PARA O DESENVOLVMENTO DE UM SISTEMA DE DISTRIBUIÇÃO E GERENCIAMENTO DE TAREFAS PARA METODOLOGIA SCRUM**

**José Antônio da Cunha**

**Cesar Analide Rodrigues**

**NATAL-RN**

RESUMO

O presente trabalho vem mostrar a utilização de algoritmos genéticos na elaboração de uma solução para um problema de remanejamento de recursos, dentro de um ambiente de desenvolvimento de software que utilizar como processo de desenvolvimento a metodologia ágil Scrum. Logo a solução proposta teve que englobar todas as nuances do processo citado e atendê-las de forma satisfatória para a geração de resultados que tivessem dentro das expectativas desejadas.

Portanto, o objetivo a ser alcançado com este projeto de pesquisa é desenvolver um sistema para distribuição e gerenciamento de tarefas para a metodologia ágil Scrum. A distribuição de tarefas vai ser baseada no tempo de execução de cada tarefa e na experiência de cada recurso. Essa distribuição vai ser feita com a utilização de uma técnica de descoberta de conhecimento em base de dados utilizando algoritmos genéticos.

**Palavras chaves**: Scrum, Algoritmos Genéticos, Processos.

# INTRODUÇÃO

Na computação, desenvolvimento de software é o ato de elaborar e implementar um sistema computacional, isto é, transformar a necessidade de um utilizador ou de um mercado em produto de software.

Nos primórdios da computação, a construção de software, tinha seu desenvolvimento informal e, esse se mostrava ineficiente. Projetos importantes apresentavam, algumas vezes, anos de atraso. O software, cujo custo superava as previsões, não era confiável, era difícil de manter e seu desempenho era insatisfatório. Esse período ficou conhecido como crise do software (SOMMERVILLE, 2007).

Diante da crise do software foi proposto, em 1968, o conceito de engenharia de software, que é um ramo da engenharia cujo foco é o desenvolvimento de software dentro dos limites impostos pelo projeto de alta qualidade (SOMMERVILLE, 2007).

O processo de software é uma das contribuições da engenharia de software para o desenvolvimento de software. Aquele pode ser definido como um conjunto de atividades e resultados associados que produz um produto de software.

Existem dois grandes grupos de processo de software (AMBLER, 2002), os prescritivos e as metodologias ágeis.

As metodologias ágeis são aquelas que seguem os conceitos propostos no manifesto ágil, que foi a idéia de um grupo de metodologistas que sugeriu melhores meios para o desenvolvimento de software e com base neste manifesto foi formulado um conjunto de princípios que definem critérios para os processos de desenvolvimento ágil (KOCH, 2005).

Dentro do grupo das metodologias ágeis uma que vem se destacando é a *Scrum*.

O Scrum é uma metodologia de gestão de projetos de software baseado em esqueleto de um processo iterativo incremental (SCHWABER, 2009).

Por ser bastante objetivo, com papeis bem definidos, de fácil adaptação e ter uma curva de aprendizado relativamente baixa, o Scrum está sendo cada vez mais utilizado como processo de desenvolvimento te software. Pesquisa realizada pela VERSIONONE em 2013 mostrou que 70% dos entrevistados utilizam o Scrum ou alguma metodologia baseada nesse.

## CONTEXTUALIZANDO ALGORITMOS GENÉTICOS

Os algoritmos genéticos são técnicas heurísticas de otimização global, ou seja, um método que encontra boas soluções a cada execução mas, não necessariamente a mesma todas as vezes.

Eles são baseados nas teorias da genética da biologia de onde tira todo o seu embasamento. Basicamente, nos algoritmos genéticos, populações de indivíduos são criadas e submetidas aos operadores de seleção, cruzamento, mutação. Estes operadores utilizam uma caracterização de qualidade de cada individuo como solução do problema em questão chamada de **avaliação** e vão gerar um processo de evolução natural destes indivíduos, que eventualmente deverá gerar um individuo que caracterizará uma boa solução, talvez até mesmo a melhor, para o problema que está sendo atacado (LINDEN, 2006).

Os algoritmos genéticos também são considerados boas técnicas para mineração de dados num processo de descobrimento de conhecimento em base de dados.

Alguns problemas que podem ser atacados com algoritmos genéticos:

* Alocação de recursos;
* Planejamento de Expansão;
* *Unit Commiment;*
* Alocação de Capacitores;
* Bioinformática

## **ALOCAÇÃO DE TAREFAS NO SCRUM COM ALGORITMOS GENÉTICOS**

A metodologia de desenvolvimento *Scrum* tem tarefas e recursos onde cada recurso tem que resolver uma tarefa dentro de um tempo determinado.

Essa configuração de problema pode ser adequada a um tipo de problema que está dentro do grupo de problemas de alocação de recursos, que é o escalonamento de tarefas.

Esse problema pode ser atacado com a implementação de um algoritmo genético, já que os problemas de alocação de recursos são áreas de aplicação desses.

## **METODOLOGIA**

A metodologia utilizada para o desenvolvimento deste projeto foi baseada em pesquisas bibliográficas em livros, revistas e artigos na área de processos de software mais especificamente o Scrum e também na área de Algoritmos Genéticos.

O estudo da metodologia ágil Scrum serviu para mostrar como é o processo de gerenciamento dentro desta metodologia e como se comportam as variáveis que estão envolvidas o processo de distribuição de tarefas, recurso, atividades e tarefas e como essas são gerenciadas no tempo em um ambiente de desenvolvimento de software

Além da pesquisa bibliográfica, foi feita uma observação do processo em ambiente real.

Houve também um estudo sobre **KDD** (*Knowledge Discovery In Databases*). Esse estudo foi voltado para a parte de mineração de dados e seus métodos, mais especificamente aos métodos baseados em algoritmos genéticos

O estudo de Algoritmos Genéticos vem como base técnica de mineração de dados a ser utilizada para o desenvolvimento do sistema. No qual esse vai, de forma aleatória, distribuir tarefas em um ambiente de desenvolvimento, de forma a apresentar uma solução onde todos os recursos e tarefas estejam relacionados, levando em consideração as nuances do processo ágil a que o sistema em questão abrange que é o *Scrum*.

# FUMDAMENTAÇÃO TEÓRICA

## PROCESSO DE DESENVOLVIMENTO DE SOFTWARE

A utilização de um processo de software tem sido apontada como um fator primordial para o sucesso de empresas de desenvolvimento de software.

Para Sommerville (SOMMERVILLE, 2007), um processo de software é um conjunto de atividades que leva à produção de um produto de software. Essas atividades requerem uma organização lógica que envolve agentes, métodos, ferramentas, artefatos, e restrições que possibilitam disciplinar, sistematizar, e organizar o desenvolvimento e a manutenção do produto de software.

Embora existam vários processos de desenvolvimento de software, existem atividades fundamentais a todos eles:

A primeira atividade é especificação de software que é a responsável pela definição das funcionalidades e das restrições do software. A segunda atividade é a de projeto e implantação do software, nesta o produto é produzido de acordo com as especificações. A terceira fase é a de validação do software. Aqui o software é avaliado para garantir que todas as funcionalidades estão de acordo com as especificações. A quarta é a evolução do software que leva ao produto correções e evoluções quando necessário ou requisitado (SOMMERVILLE, 2007).

Os processos de software podem ser divididos em dois grupos: processos prescritivos e processos ágeis.

No grupo dos processos prescritivo estão incluídos: o modelo cascata, o evolucionário, a prototipação, entre outros. Estes tipos de processos, também conhecidos como pesados ou orientados a documentação, são mais indicados em situações em que os requisitos são estáveis e os requisitos futuros são previsíveis. Estas situações são difíceis de serem atingidas, uma vez que os requisitos para o desenvolvimento de um software são mutáveis. Dentre os fatores responsáveis por alterações nos requisitos estão à dinâmica das organizações e as mudanças pedidas pelos *stakeholders.* (SOARES, 2008).

O termo “metodologias ágeis” se tornou popular quando um grupo de especialista em processos de software definiu um manifesto para encorajar melhores meios de desenvolver softwares e, com base nesse manifesto, formulou um conjunto de princípios que definem critérios para o desenvolvimento ágil de software. (AMBLER, 2002).

Os valores contidos nesse manifesto definem preferências, que encorajam o enfoque de certas áreas, mas não elimina as outras. Esses valores são, segundo Ambler:

* Indivíduos e interações valem mais do que processos e ferramentas;
* Um software funcionando vale mais do que documentação extensa;
* A colaboração vale mais que a negociação de contrato;
* Responder a mudanças mais do que seguir um plano. (AMBLER, 2002, pg. 24)

No grupo de metodologias ágeis estão: Extreme Programming (XP), Scrum, DSDM, Crystal e outros. Essas metodologias seguem os valores e princípios do manifesto ágil buscando uma maior iteração com o cliente, se adaptando rapidamente a mudanças e procurando dar uma maior ênfase na entrega de um produto de software e confeccionando apenas a documentação necessária.

Os processos e metodologias têm suas diferenças e cada um tem sua área de aplicação, cabe a equipe desenvolvimento pesquisar qual o seu foco e como deve ser desenvolvido o seu projeto e escolher entre os processos e metodologias e começar o seu desenvolvimento.

Neste trabalho a metodologia a ser abordada mais profundamente será a metodologia ágil Scrum, que será mais bem mostrada no próximo capítulo.

## **SCRUM**

Métodos, práticas e técnica para o desenvolvimento ágil de projetos prometem aumentar a satisfação do cliente (BOEHM, 2003) para produzir alta qualidade de software e acelerar os prazos de desenvolvimento de projetos. Entre as metodologias ágeis destacam-se a Extreme Programming (XP), *Dynamic System Development Method* (DSDM), *Crystal, Lean Development* e *SCRUM.*

Uma pesquisa realizada pela versionone em 2007 mostrou que 40% dos entrevistados usam SCRUM puro nos seus projetos e outros 60% usam uma metodologia híbrida com o Scrum.

Segundo Schwaber (SCHWABER, 2004), o SCRUM é uma metodologia ágil que se molda sobre um esqueleto de um processo iterativo e incremental. Ele é usado em trabalhos complexos nos quais não é possível prever tudo que irá ocorrer e oferece um framework que torna tudo visível. Isso permite aos praticantes do SCRUM saber exatamente o que está acontecendo ao longo do projeto e fazer os devidos ajustes para manter o projeto se movendo ao longo do tempo visitando alcançar os objetivos.

O SCRUM pode ser divido nas seguintes fases *pregame*, *game*, e *postgame*, segundo Schwaber.

No pregame está o planejamento, que serve para definir o que será feito junto com as estimativas de tempo e custo. Esta fase, fazendo uma comparação com uma metodologia iterativa e incremental, consiste na concepção e elaboração. Nessa fase é montado o *product* *backlog*, lista de funcionalidades, e aqui serão priorizadas funcionalidades que entrarão na etapa de desenvolvimento.

No game está o desenvolvimento das funcionalidades que foram priorizadas na fase anterior, seguindo sempre o que foi determinado com relação a tempo, custo, qualidade. A figura 1, ilustra o clico de progresso do Scrum.

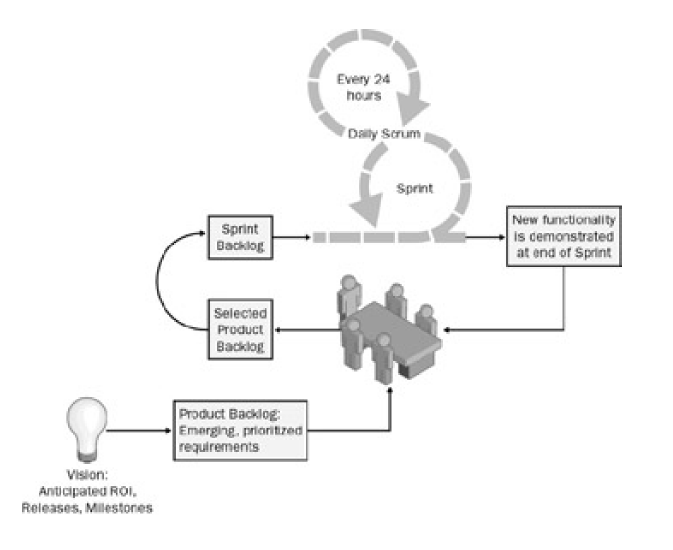


Figura 1: Workflow do processo Scrum

Fonte SCHWABER, 2004

Estas fases estão bem divididas no tempo, o seu progresso é baseado em iterações, cada uma com duração de 2 a 4 semanas, chamadas de Sprint. Antes de cada um, se realiza uma reunião de planejamento onde a equipe de desenvolvedores tem contato com o cliente para priorizar o trabalho a ser feito, selecionar e estimar as tarefas que o time pode realizar dentro dele.

Depois desses passos se dá inicio a execução do Sprint, durante esta fase a equipe controla o andamento do desenvolvimento realizando reuniões diárias que não têm duração maior do que 15 minutos, e a observando o progresso usando várias ferramentas entre elas o *Sprint Burndown*. Ao final de cada Sprint é feita uma revisão do produto entregue para verificar se tudo está dentro do que foi colocado para o desenvolvimento.

Ao fechamento do Sprint é realizada uma reunião de revisão, onde o a equipe demonstra o produto gerado, e o valida com o cliente, se o objetivo foi atingido. Logo depois é realizada uma reunião de retrospectiva para avaliar o aprendizado com o objetivo de melhorar o processo/time e o produto para o próximo Sprint.

Segundo Pereira (PEREIRA, 2009), o Scrum torna-se ideal para projetos dinâmicos e suscetíveis a mudança de requisitos, sejam eles novos ou apenas modificados. No entanto é preciso entender seus papeis, responsabilidades, artefatos e conceitos para uma aplicação satisfatória da metodologia.

## PAPEIS

Existem apenas três papéis no SCRUM: o *Product Owner* (Cliente), o time (equipe de desenvolvimento) e o *Scrum Master*. Todas as responsabilidades gerenciais são dividas entre esses três papeis (SCHWABER, 2004).

O *Product Owner* é responsável por representar todos os interessados no projeto e no sistema resultante, ele também e responsável pelo financiamento do projeto, pelo retorno financeiro do produto, elaborar o plano de entregas, priorizar os requisitos, essa priorização é para assegurar que as funcionalidades mais importantes sejam desenvolvidas primeiro. O *Product Owner* também faz avaliação do resultado no fim da iteração e aceita ou rejeito o que foi apresentado (SCHWABER, 2004).

Como fazer dos dados que as empresas guardam sobre seus clientes e sobre ela mesma, se transformem em informações para que a empresa possa aprender o comportamento daqueles e como ela está se comportando em relação e eles, para que possa evoluir e dar um melhor atendimento e como colocar a disposição produtos e/ou serviços que

O *Scrum Team* é responsável pelo desenvolvimento do produto. Ele é alto-organizado, auto gerenciado e multifuncional. A equipe contém entre 5 e nove membros e todos eles têm responsabilidade direta sobre os resultados obtidos em cada iteração e no projeto como um todo (PEREIRA, 2009).

O *Scrum Master* desempenha um papel de liderança, gerenciando os interesses do *Product Owner* mediante o *Scrum Team*. Além dessa mediação ele é responsável pelo processo como um todo, tendo que tutorar todos os envolvidos no projeto para seguirem todos os passos da metodologia (SCHWABER, 2004). Garantir que o time esteja totalmente funcional, eliminar as dificuldades que estejam atrapalhando o desenvolvimento da iteração e proteger a equipe de interferências externas também são responsabilidades do *Scrum Master* (PEREIRA, 2009).

As pessoas que representam esses papéis são aquelas que estão diretamente envolvidas com o projeto. Muitos outros estarão interessados no projeto, mas não influenciarão diretamente do desenrolar desse. O Scrum faz uma clara distinção entre estes dois grupos e assegura que aqueles que têm responsabilidade direta sobre o projeto tenham total autoridade para fazer o que for necessário para que se atinja os resultados desejados e aqueles que não têm uma interação direta não interfiram desnecessariamente (SCHWABER, 2004).

## **ARTEFATOS**

Segundo Schwaber (SCHWABER, 2004), o *Scrum* apresenta poucos artefatos que são: o *Product Backlog* e o *Sprint Backlog*.

O *Product Backlog* contém uma lista de requisitos que são necessários para a concepção de um sistema/produto que possam ser associados com o valor de negócio (SCHWABER, 2004). É importante que cada item do *Product Backlog* ter um valor de negocio agregado para a finalização do projeto (PEREIRA, 2009).

O responsável pelo conteúdo do *Product Backlog* é o *Product Owne*r, este também é responsável pela priorização e pela viabilidade do artefato.

Um exemplo de *Product Backlog* pode ser visto na tabela 1 a abaixo:

Tabela 1: Exemplo de *Product Backlog*

|  |  |  |
| --- | --- | --- |
| Funcionalidade | Prioridade | Sprint |
| Requisito 01 | 1 | 1 |
| Requisito 02 | 2 | 1 |
| Requisito 03 | 2 | 1 |
| Requisito 04 | 3 |  |
| Requisito 05 | 4 |  |
| Requisito 06 | 4 |  |

Fonte: SCHWABER, 2004

Existem muitas formas para gerenciar o *Product Backlog*, todas elas precisam que cada item seja identificado e estimado em tempo ou tamanho, e que sua ordem de importância seja estabelecida pelo *Product Owner*. Com esses atributos é possível ter os itens da lista em uma ordem de priorização (PEREIRA, 2009).

Outro artefato do *Scrum* é *Sprint Backlog*, este é uma lista de que define ao trabalho, ou as tarefas que a equipe seleciona dentro do *Produtct Backlog*, que são um conceito do *Scrum* chamado “Incremento de funcionalidade potencialmente implementável”. Uma funcionalidade potencialmente implementável significa que quando esse item for finalizado ele estará completamente funcional e resolvendo algum problema de negócio do projeto (SCHWABER, 2004).

A montagem do *Sprint Backlog* é feita na *Sprint Planning Meeting* (Reunião de planejamento de Sprint). Aqui os itens do *Product Backlog* são estimados pela equipe em conjunto com o *Scrum Master* e *Product Owner*. Uma técnica muito interessante e objetiva conhecida como *Planning Poker* pode ser feita em Horas/Tamanho.

O *Planning Poker* funciona como uma forma de estimativa em conjunto, pode ser feita como um jogo. Todos os envolvidos diretamente participam de forma democrática para chegar a um consenso de estimativa, para cada item do Product Backlog, de forma objetiva e divertida (PEREIRA, 2009).

As tarefas devem ser dividas para que cada uma tenha um tempo de resolução de 4 a 16 horas. Funções com tempo maior que 4 a 16 horas são consideradas meros espaços para tarefas que ainda não foram bem estimadas (SCHWABER, pg. 24, 2004).

Apenas o time pode fazer mudanças no Sprint *Backlog*, esse é altamente visível e está sempre mostrando ao time o que precisa ser feito.

Tabela 2: Sprint Backlog

|  |  |  |  |
| --- | --- | --- | --- |
| Funcionalidade | Prioridade | Alocado | Pronto |
| Requisito 1 | 1 | Recurso 1 | X |
| Requisito 2 | 1 | Recurso 2 |  |
| Requisito 3 | 1 | Recurso 3 |  |

Fonte: SCHWABER, 2004

## SPRINT

O *Sprint* é onde ocorre o progresso do projeto no Scrum aquele tem uma duração entre 2 a 4 semanas chamada de Time Box.

O Scrum é baseado em entrega de produto a cada Sprint, logo é recomendável que durante o desenvolvimento do projeto exista um padrão, um mesmo Time Box, para todos os *Sprints*. Isso ajudará a medir o progresso e a produtividade da equipe no projeto (PEREIRA, 2009).

Durante o *Sprint*, o time, de forma organizada, controla coma as tarefas devem ser executadas. Aqui, a equipe não deve sofrer interferências externas, e o responsável por evitar essa interferência é o Scrum Máster. O acompanhamento do progresso realizando reuniões diárias (*Dayly Meeting*). Essas reuniões são feitas pelo time juntamente com o Scrum Máster e a equipe trabalha apenas três questões:

* + O que foi feito desde ontem?
  + O será feito Hoje?
  + Existe alguma dificuldade para o seu trabalho?

Todo e qualquer problema encontrado durante a reunião deve ser tratado em outra reunião apenas com os envolvidos (PEREIRA, 2009).

Ao final do Sprint, o time entrega o produto testado e revisado realizando uma demonstração prática. Este é o momento do *Product Owner* inspecionar o produto final e verificar se o mesmo está de acordo com o que foi pedido. Essa apresentação se da em uma reunião chamada de *Sprint Review*. Depois da apresentação a equipe faz uma avaliação do trabalho feito, para observar os erros e acertos para estabelecer melhorias para o próximo Sprint (PEREEIRA, 2008).

Usar *Scrum* nos projetos ajuda a construir somente o que o cliente valoriza e não mais que isso criando melhor adaptados à realidade do cliente, além de as práticas desta metodologia trazer vantagens ao gerenciamento de projetos como a participação mais efetivada da equipe e do cliente, maior visibilidade do que está sendo feito, incentivo ao compartilhamento e disseminação de conhecimento entre outros.

# **DESCOBERTA DE CONHECIMENTO EM BASE DE DADOS**

A análise de grandes quantidades de dados pelo homem é inviável sem o auxílio de ferramentas computacionais apropriadas. Portanto segundo (GOLDSHMIDT, PASSOS, 2005), torna-se imprescindível o desenvolvimento de ferramentas que auxiliem o homem, de forma automática e inteligente, na tarefa de analisar, interpretar e relacionar esses dados para que possa desenvolver e selecionar estratégias de ação em cada contexto de aplicação.

Para atender a essa necessidade, surge uma nova área denominada Descoberta de conhecimento em Bases de Dados (*Knowledge Discovery in Databases - KDD*).

Segundo (GOLDSHMIDT, PASSOS, 2005) O termo KDD foi formalizado em 1989 em referência ao amplo conceito de procurar conhecimento a partir de bases de dados. Uma das definições mais populares foi proposta por um grupo de pesquisadores: “KDD é um processo, de varias etapas, não trivial, interativo e iterativo, para identificação de padrões compreensíveis, validos, novos e potencialmente úteis a partir de grandes conjuntos de dados”.

Para (GOLDSHMIDT, PASSOS, 2005), a Descoberta de Conhecimento em Base de Dados é caracterizada como um processo composto por várias etapas operacionais que são organizadas em três grupos que são: pré-processamento, mineração de dados e pós-processamento.

A etapa de Pré-Processamento tem como objetivo preparação dos dados para os algoritmos da etapa seguinte, a Mineração de Dados. Durante esta etapa, é realizada a busca efetiva por conhecimentos úteis no contexto da aplicação de KDD. A etapa de Pós-Processamento abrange o tratamento do conhecimento obtido na Mineração de Dados. Tal tratamento, nem sempre é necessário, tem a finalidade de viabilizar a avaliação da utilidade do conhecimento. (GOLDSHMIDT, PASSOS, 2005, pg.3)

De uma maneira geral, a complexidade do processo de KDD está na dificuldade em perceber e interpretar adequadamente inúmeros fatos observáveis durante o processo e na dificuldade em conjugar dinamicamente tais interpretações de forma a decidir quais ações devem ser realizadas em cada caso (GOLDSCHMIT, 2003). Cabe ao analista humano a árdua tarefa de orientar a execução do processo de KDD.

Segundo (GOLDSHMIDT, PASSOS, 2005), as atividades na área de KDD podem ser organizadas em três grandes grupos:

* Desenvolvimento Tecnológico – esse item abrange todas as iniciativas de concepção, aprimoramento e desenvolvimento de algoritmos, ferramentas e tecnologias de apoio que possam ser utilizados na busca por novos conhecimentos em grandes bases de dados.
* Execução de KDD – Esse item refere-se às atividades voltadas à busca efetiva de conhecimento em bases de dados. As ferramentas produzidas pelas atividades de desenvolvimento tecnológico são utilizadas na execução do processo de KDD.
* Aplicação de Resultados – Finalmente, uma vez obtidos modelos de conhecimentos úteis a partir de grandes bases de dados, as atividades se voltam à aplicação dos resultados no contexto em que foi realizado o processo de KDD. (GOLDSHMIDT, PASSOS, 2005, pg. 6)

Para (GOLDSHMIDT, PASSOS, 2005), A expressão técnica de KDD se refere a qualquer teoria que possa fundamentar a implementação de um método de KDD.

Existem diversos tipos de técnicas e de algoritmos para a mineração de dados. Esses Podem ser subdivididos em: Técnicas Tradicionais, Técnicas Especificas e Técnicas Híbridas.

* Técnicas tradicionais são tecnologias que existem independentes do contexto da Mineração de Dados. Em geral, produzem bons resultados também em aplicações de KDD. Como exemplo de tecnologias tradicionais pode-se citar: Redes Neurais, Lógica Nebulosa, Algoritmos Genéticos e estatística.
* Técnicas especificas, são técnicas desenvolvidas especificamente para aplicação em tarefas de KDD.
* Técnicas Hibridas são aquelas que utilizam mais de uma técnica para a solução de um problema de modelagem. A grade vantagem desse tipo de sistema deve-se ao sinergismo obtido pela combinação de duas ou mais técnicas de modelagem. Esse se reflete na obtenção de um sistema mais poderoso e com menos deficiências. (GOLDSHMIDT, PASSOS, 2005 pg. 18),

Segundo (GOLDSHMIDT, PASSOS, 2005), a necessidade de manipulação de grandes e heterogêneos volumes de dados, o tratamento de resultados representados em diferentes formatos e a dificuldade de integração de diversos algoritmos específicos são alguns exemplos de fatores operacionais. Atualmente, encontram-se comercialmente disponíveis diversas ferramentas que programam ambientes integrados para facilitar a execução das etapas operacionais de KDD, minimizando as dificuldades decorrentes dos fatores operacionais. Em geral essas ferramentas reúnem diverso métodos de Mineração de Dados podendo ser aplicadas em várias tarefas.

Para (GOLDSHMIDT, PASSOS, 2005), o papel do usuário dos processos de KDD é de grande importância e essa participação humana deve ser especializada.

De uma forma geral cabe ao analista humano a árdua tarefa de orientar a execução de processo de KDD. Para tanto, diante de cada cenário, o homem utiliza sua experiência anterior, seus conhecimentos e sua intuição para interpretar e combinar subjetivamente os fatos de forma a decidir qual estratégia deve ser adotada. Os cenários envolvem, em geral, inúmeros aspectos tais como: fatos observados cuja origem e os níveis de detalhe são os mais diversos e difusos, resultados intermediários obtidos com várias ferramentas, opiniões de especialistas no contexto da aplicação e conhecimento previamente existente.

No entanto, a formação de especialistas em KDD constitui-se em uma tarefa longa e exaustiva, pois requer não somente uma fundamentação teórica sobre a área, mas também a participação destes em inúmeras experiências práticas reais.

## **MINERAÇÃO DE DADOS**

A Mineração dos dados é uma das etapas do processo de KDD. Nela é efetuada a busca efetiva por conhecimentos úteis no contexto da aplicação de KDD (GOLDSHMIDT, PASSOS, 2005. Apud, FAYYAD, 1996).

Para a tarefa de mineração de dados são utilizados métodos de mineração de dados esses métodos podem ser considerados operadores definidos a partir de precondições e efeitos. Uma precondição é um predicado que estabelece um requisito que deve ser cumprido antes da execução do método. Um efeito também é um predicado que descreve uma situação gerada após a aplicação do método (GOLDSHMIDT, PASSOS, 2005).

O método de mineração é baseado em algum algoritmo um deles é o Algoritmo Genético que é extremamente útil em problemas que envolvam otimização (GOLDSHMIDT, PASSOS, 2005).

# **ALGORITMOS GENÉTICOS.**

Para (LINDEN, 2006), os Algoritmos Genéticos fazem parte do conjunto dos Algoritmos Evolucionários. Esses usam modelos computacionais dos processos naturais de evolução como uma ferramenta para resolver problemas.

Os algoritmos evolucionários funcionam mantendo uma população de estruturas, denominadas indivíduos ou cromossomos, que se comportam de forma semelhante às evoluções das espécies. A estas estruturas, são aplicados operadores genéticos, como recombinação e mutação entre outros. Cada indivíduo, recebe uma avaliação, que é uma quantificação de sua qualidade, como solução para um problema em questão. Com base nesta avaliação, serão aplicados os operadores genéticos, de forma a simular a sobrevivência do mais apto (LINDEN, 2006).

Os algoritmos evolucionários, se posicionam como técnicas de busca. Eles, são parte de um ramo de busca, chamado de “Técnicas Aleatórias Guiadas”, isto é, eles têm componentes aleatórios, mas usam as informações do estado corrente, para guiar esta pesquisa. Como se pode ver na figura 2, os algoritmos evolucionários se dividem em vários tipos distintos.

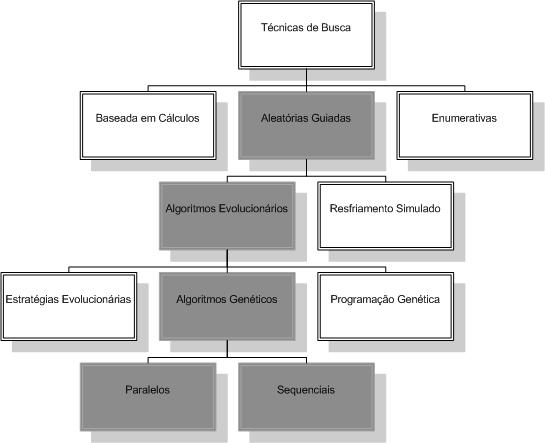


Figura 2: Hierarquia de Técnicas de Busca

Fonte: LINDEN, 2004

Os algoritmos genéticos (GA), são um ramo dos algoritmos evolucionários e, como tal, podem ser definidos, como uma técnica de busca, baseada em uma metáfora do processo biológico de evolução natural.

Segundo (GOLDSHMIDT, PASSOS, 2005), são modelos computacionais de busca e otimização de soluções em problemas complexos. São técnicas, que procuram obter boas soluções, por meio da evolução de populações de indivíduos que foram criados.

Para (LINDEN, 2006), os indivíduos depois de criados, são submetidos aos operadores genéticos de recombinação e/ou mutação. Estes operadores, utilizam uma caracterização de qualidade de cada indivíduo, como solução do problema em questão, chamada de avaliação e, vão gerar um processo de evolução natural destes indivíduos, que eventualmente, deverá gerar um indivíduo, que caracterizará uma boa solução para o problema.

Os Algoritmos Genéticos, empregam um processo adaptativo e paralelo de busca de soluções em problemas complexos. O processo é adaptativo, pois as soluções existentes a cada instante, influenciam a busca por futuras soluções. O paralelismo do processo é decorrência natural do fato de que diversas soluções são consideradas a cada momento pelos GA (GOLDSHMIDT, PASSOS, 2005).

Para (LINDEN, 2006), os GA são técnicas heurísticas, ou seja, eles não encontram a solução ótima para um problema, e quando o faz, nem sempre consegue repetir o feito. Eles não ficarão estagnados pelo fato de terem encontrado a solução melhor possível. Nesse ponto, os GA se parecem com a evolução natural, que pelo fato de ter encontrado um indivíduo, que é instantaneamente o melhor de certo grupo, não para de procurar outros indivíduos ainda melhores. Na evolução natural, isto também decorre, de circunstâncias que mudam de um momento para o outro. Uma bactéria pode ser a melhor em um ambiente livre de antibióticos, mas quando esses são usados, outras que antes eram menos fortes tornam-se as únicas sobreviventes por serem as únicas adaptadas.

No caso dos algoritmos genéticos, o ambiente normalmente é um só. Entretanto, conforme as gerações vão se passando e, os operadores genéticos vão atuando, faz-se uma grande busca pelo espaço de soluções, busca esta que seria realizada pela evolução natural se elas ficassem permanentemente em um ambiente imutável.

Mas, ao contrário do que se possa pensar, o processo de evolução natural não é dirigido à obtenção da solução ótima. Na verdade, o processo simplesmente consiste em, fazer competir uma série de indivíduos e, pelo processo de sobrevivência do mais apto, os melhores indivíduos tendem a sobreviver. Um GA tem o mesmo comportamento que a evolução natural: a competição entre os indivíduos, é o que determina as soluções obtidas. Eventualmente, devido à sobrevivência do mais apto, os melhores indivíduos prevalecerão.

Sendo assim, Para (LINDEN, 2006), apesar de seu nome implicar no contrario, os GA não consistem algoritmos de busca da solução ótima de um problema, mas sim uma heurística que encontra boas soluções a cada execução, mas não necessariamente, a mesma, todas às vezes.

Segundo (GOLDSHMIDT, PASSOS, 2005), os GA, em geral, são compostos pelos seguintes componentes: problema, representação do cromossomo, avaliação, seleção, operadores genéticos e técnicas e parâmetros.

Para (LINDEN, 2006), se pode dizer que os Algoritmos Genéticos, são técnicas de busca com as seguintes características positivas:

* Paralela – pois mantém uma população de soluções que são avaliadas simultaneamente.
* Global – os Algoritmos Genéticos não usam apenas informação local, logo, não necessariamente ficam presos em máximos locais como certos métodos de busca. Esta característica é uma das mais interessantes dos GA e faz com que eles sejam uma técnica extremamente adequada para funções multidimensionais e de perfis complexos como a maioria das funções de custo associadas a problemas reais.
* Não totalmente aleatória – existem métodos que usam apenas variáveis aleatórias para realizar suas pesquisas. GA têm componentes aleatórios, mas como usam as informações da população corrente para determinar o próximo estado da busca, não podem ser considerados totalmente aleatórios.
* Não afetada por descontinuidades na função ou em suas derivadas – Os GA não usam informações de derivadas na sua evolução nem necessitam de informação sobre o sei entorno para poder efetuar sua busca. Isto faz com que sejam muito adequados para funções com descontinuidades ou para as quais não temos como calcular uma derivada.
* Capaz de lidar com funções discretas e continuas – os GA são capazes de lidar com funções reais, discretas, booleanas e até mesmo categóricas, sendo possível inclusive misturar as representações sem prejuízo para a habilidade dos GA de resolver problemas (LINDEN, 2006, pg. 49).

Tendo em vista que, são buscas direcionadas e inteligentes, Algoritmos Genéticos, são boas técnicas, para atacar problemas de busca com espaços muito grandes, que não podem ser resolvidos por técnicas tradicionais. Como exemplo de problemas para aplicação de Algoritmos genéticos têm-se: Otimização de funções matemáticas, otimização combinatorial, otimização de planejamento, problema do caixeiro viajante, problema de otimização de rota de veículos, otimização de distribuição e logística, seleção de variáveis em mineração de dados, dentre outros.

## **GA X GENÉTICA/EVOLUÇÃO DAS ESPÉCIES**

A terminologia, é muito importante, já que, os GA são inspirados na genética e na teoria da evolução das espécies, ocorrendo, uma forte analogia entre os termos desses dois ambientes.

Na genética, os cromossomos, se combinam para formar as características do individuo, já na área dos GA os cromossomos, são estruturas de dados que formam indivíduos que representam uma possível solução para um problema.

No campo da genética, os cromossomos são formados por genes, que são cada unidade de informação hereditária presente no cromossomo, genes que podem ter determinado valor, são chamados de alelos, e sua posição no cromossomo de lócus. No GA usam-se os termos característica, valor e posição respectivamente para as terminologias apresentadas acima.

Além da terminologia, existem outras funções que os algoritmos genéticos e a biologia compartilham, tais como: população, mutação, cruzamento ou recombinação e seleção.

## **ESTRUTURA DO GA**

O algoritmo genético pode ser resumido algoritmicamente através dos seguintes passos?

1. Inicialize a população de cromossomos
2. Avalie cada cromossomo na população
3. Selecione os pais para gerar novos cromossomos
4. Aplique operadores genéticos
5. Apague os velhos membros da população
6. Avalie todos os novos cromossomos e insira-os na população
7. Se a condição de parada retorne o melhor individuo da população, se não retorne ao passo c (LINDEN, pg. 53, 2006).

Esta é apenas uma visão de alto nível, entretanto, como o GA é altamente genérico, vários de seus componentes são invariáveis de um problema para o outro. Isto favorece a sua implementação em uma linguagem orientada a objetos, permitindo o reaproveitamento do código para a solução de vários problemas diferentes (LINDEN, 2006).

## **OBTENDO SOLUÇÕES COM ALGORITMOS GENÉTICOS**

Para que o algoritmo genético funcionar existe alguns aspectos que devem ser avaliados e especificados para que o algoritmo tenha o seu desempenho satisfatório. Estes aspectos são: a representação cromossomial adequada ao problema e uma função de avaliação que avalie satisfatoriamente o grau de adequação de cada individuo como solução do problema.

### REPRESENTAÇÃO DO CROMOSSOMO

A representação do cromossomo é fundamental para o GA. Basicamente, ela consiste em uma maneira de traduzir a informação do nosso problema em uma maneira viável de ser tratada pelo computador. Quanto mais ela for adequada ao problema, maior a qualidade dos resultados obtidos (LINDEN, 2006).

Cada pedaço indivisível desta representação desta representação é chamado de gene, por analogia com as partes fundamentais que compõe as partes fundamentais de um cromossomo biológico.

Segundo LINDEN (2006), é importante notar que a representação do Cromossomo é completamente arbitrária ficando a sua definição de acordo com o gosto do desenvolvedor e com a adequação ao problema. È interessante apenas que algumas regras gerais sejam seguidas:

1. A representação deve ser a mais simples possível
2. Se houver soluções proibidas para o problema, então elas não devem ter uma representação.
3. Se o problema impuser condições de algum tipo, estas devem estar implícitas dentro da representação (LINDEN, 2006, pg. 54).

### FUNÇÃO DE AVALIAÇÃO

A função de avaliação é a maneira pela utilizada pelo GA para determinar a qualidade de um individuo como solução de um problema a ser resolvido. Pode-se entende-la mais facilmente se a visualizar como uma nota dada ao individuo na resolução do problema. Esta nota será usada para a escolha de indivíduos pelo módulo de seleção, sendo a forma de diferenciar entre as boas e as más soluções para um problema (LNDEM, 2006).

Dada a generalidade dos GA, a função de avaliação, em muitos casos, é a única ligação verdadeira do sistema como o problema real. Isto decorre do fato que a função de avaliação só julga a qualidade da solução que está sendo apresentada por aquele individuo, sem armazenar qualquer tipo de informação sobre as técnicas de resolução do problema.

O mais importante conceito a ter em mente é que a função de avaliação deve refletir os objetivos a serem alcançados na resolução do problema e é derivada diretamente das condições impostas por esse (LINDEN, 2006).

### POPULAÇÃO

Nos Algoritmos Genéticos, a População representa o conjunto de soluções geradas por ele.

O tamanho da população determina o numero de soluções que o GA gera, a cada iteração de busca, até que uma condição seja satisfeita. Este tamanho afeta diretamente o desempenho global e a eficiência do algoritmo (VALENTIM, 2008). Caso o número de indivíduos na população seja muito pequeno, não haverá espaço para uma variedade genética suficientemente grande o que fará com que o GA seja incapaz de achar boas soluções e caso o número seja muito grande o algoritmo demorará muito e se pode estar aproximando-se de uma busca exaustiva.

### SELEÇÃO DOS PAIS

O método de seleção de simula o mecanismo de seleção natural que atua sobre as espécies biológicas, em que os pais mais capazes geram mais filhos, ao mesmo tempo em que os pais menos aptos também conseguem gerar descendentes (LINDEN, 2006).

Tem que se privilegiar os indivíduos com função de avaliação alta, sem desprezar aqueles indivíduos com função de avaliação extremamente baixa. Esta se dá, pois, mesmo esses indivíduos podem ter características genéticas que sejam favoráveis à criação de um individuo que seja a melhor solução para o problema que está sendo atacado.

### VANTAGENS E DESVANTAGENS DOS ALGORITMOS GENÉTICOS

Os algoritmos genéticos são uma ferramenta para efetuar buscas em espaços muito grande de soluções e suas principais vantagens são:

* São métodos robustos, apresentando bons resultados em espaços de busca complexos.
* É uma técnica que estabelece uma analogia com a natureza, são simples de serem implementados, pois não é necessário o conhecimento matemático profundo sobre o problema.
* Trabalham com todo o conjunto de soluções gerado e não apenas com um ponto o que permite realizar buscas em vários espaços.
* Apresenta bons resultados na resolução de problemas de otimização de grande escala.
* Utiliza regras de transação probabilística ao invés de determinística.

Mas os algoritmos genéticos não são a solução milagrosa para todos os problemas computacionais existentes. Na verdade eles se limitam a problemas como espaços de busca muito grandes. Ainda existem desvantagens como:

* Dificuldade em se encontrar a solução ótima.
* Necessidade de varias avaliações de valores de aptidão
* Dificuldade na escolha da representação da solução (Cromossomo).

Por essas características é importante saber se o seu problema pode ser resolvido por um algoritmo genético, e não adaptar o aquele a uma forma de algoritmo genético, pois assim sua solução pode ser prejudicada.

# **PROBLEMA DO ESCALONAMENTO DE TAREFAS**

O problema de escala de tarefas consiste em fazer uma escala dessas, onde cada tarefa consiste em um seqüência de operações, que têm que ser processadas em um conjunto fechado e limitado de maquinas, de forma que todas as tarefas sejam executadas em um tempo mínimo (LINDEN, 2006).

Este problema foi adaptado a realidade de um projeto de desenvolvimento de software seguindo o processo Scrum, ou seja, no Sprint, onde estão situados os recursos que irão desenvolver e as tarefas (funcionalidades) que serão desenvolvidas, como distribuir as tarefas para os recursos para que:

* Todos os recursos tenham tarefas
* Todas ou a maioria de tarefas estejam dentro do Sprint Para serem desenvolvidas
* Gerenciar esse desenvolvimento

Baseado nesse problema foi desenvolvido um sistema de distribuição e gerenciamento de tarefas para o processo Scrum para atender essas funcionalidades.

# **SISTEMA DE DISTRIBUIÇÃO DE TAREFAS BASEADO EM SCRUM**

O sistema de distribuição de tarefas é uma ferramenta capaz de gerar uma sugestão que tarefas devem ser alocadas a recursos, para que aquelas sejam realizadas, baseado na capacidade de resolução de tarefas do recurso (experiência), e o tempo que cada tarefa tem para ser realizada. Além da distribuição de tarefas, o sistema também oferece um módulo de cadastro e um módulo de gerenciamento.

O módulo de cadastro e responsável pelo cadastro de projetos, recursos, tarefas e *Sprints*, para a população de dados para a distribuição de tarefas.

O módulo de gerenciamento serve para auxiliar o usuário de como anda o andamento do desenvolvimento das tarefas, e dá suporte para a avaliação da capacidade de resolução das tarefas por parte do recurso e a estimativa das tarefas.

## DESENVOLVIMENTO DO PROJETO

### PROCESSO DE DESENVOLVIMENTO

O processo de desenvolvimento utilizado para o projeto do sistema de distribuição de tarefas foi o ICONIX. A escolha se deu por esse processo ser relativamente pequeno, simples, conduzido por casos de uso e iterativo e incremental além de usar a UML como linguagem de modelagem.

### AMBIENTE DE DESENVOLVIMENTO

O ambiente de desenvolvimento que foi utilizado para a produção do sistema e realização de teste está descrito nas tabelas3 e 4.

Tabela 3: Recurso de Hardware

|  |  |
| --- | --- |
| **Descrição** | **Qtd** |
| Microcomputador PC para desenvolvimento e testes. Configuração: Processador Core 2 Du, 2.43 GHz, 4GB RAM, 500 GB HD, com conexão a internet | 01 |

Fonte: Elaborada pelo Autor

Tabela 4: Recursos de Software

|  |  |
| --- | --- |
| **Descrição** | **Qtd** |
| Sistema operacional Windows XP | **01** |
| Linguagem de Programação C# |  |
| Ferramenta de Modelagem UML Jude Community 5.3.1 | **01** |
| Ferramenta de desenvolvimento IDE Visual Studio 2008 | **01** |
| Sistema Gerenciador de Banco de Dados SQL SERVER 2005 | **01** |

Fonte: Elaborada Pelo Autor

### ANÁLISE DO PROJETO

A análise para o projeto do sistema foi feita seguindo uma seqüência de passos onde o primeiro foi o levantamento de requisitos, a modelagem do projeto em UML onde foram gerados o diagrama de casos de uso e o diagrama de domino do projeto.

No levantamento dos requisitos foram elencados os que compõem a tabela 5, esses todos funcionais:

Tabela 5: Requisitos Funcionais do Sistema

|  |  |  |  |
| --- | --- | --- | --- |
| **Código** | **Nome** | **Descrição** | **Categoria** |
| F01 | Manter Grandezas do processo Scrum | Esse requisito consiste em fazer a manutenção, ou seja, as operações de Inserção, Remoção, Edição e listagem das grandezas do processo Scrum, que são necessárias para p sistema de distribuição de tarefas. Estas grandezas são: *Sprint,* Recursos e tarefas. | Evidente |
| F02 | Distribuir Tarefas | Principal requisito do sistema, no qual as tarefas serão dividas entre os recursos para que todas as atividades estejam associadas a recursos capazes de resolvê-las. | Evidente |
| F03 | Gerenciar Tarefas e Atividades | Este requisito tem por finalidade o gerenciamento de execução das tarefas por parte dos recursos, e daí fazer uma avaliação do recurso e da estimativa da tarefa. | Evidente |

Fonte: Elaborada Pelo Autor

Além dos requisitos funcionais que foram listados na tabela 5 acima foram elencados, também, os requisitos não funcionais que são restrições de hardware e de software, atendimento de qualidade, manutenibilidade, usabilidade e desempenho do produto, ilustrado na tabela 6.

Tabela 6: Requisitos não funcionais do Sistema

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Código | Nome | Descrição | Categoria | Tipo |
| NF01 | Utilização de Algoritmos | O sistema deve fazer a distribuição das tarefas, de forma aleatória utilizando um método de mineração de dados baseado em algoritmo genético | Evidente | Fixo |
| NF02 | Linguagem de desenvolvimento | A linguagem de desenvolvimento a ser utilizada deve ser a C# | Evidente | Fixo |

. Fonte: Elaborada Pelo Autor

Depois de definir os requisitos funcionais do sistema, foi feita a análise desses, e conseqüentemente feita uma modelagem UML que gerou alguns diagramas que vão servir de base para o desenvolvimento do sistema. O primeiro diagrama gerado foi o diagrama de casos de uso do sistema. Esse diagrama mostra as ações o que o sistema deve realizar para que seja atendido um requisito.

A figura 3 mostra o diagrama de casos de uso obtido a partir da análise dos requisitos, e a figura 4, ilustra a descrição dos casos de uso, ou seja, seus detalhamentos.

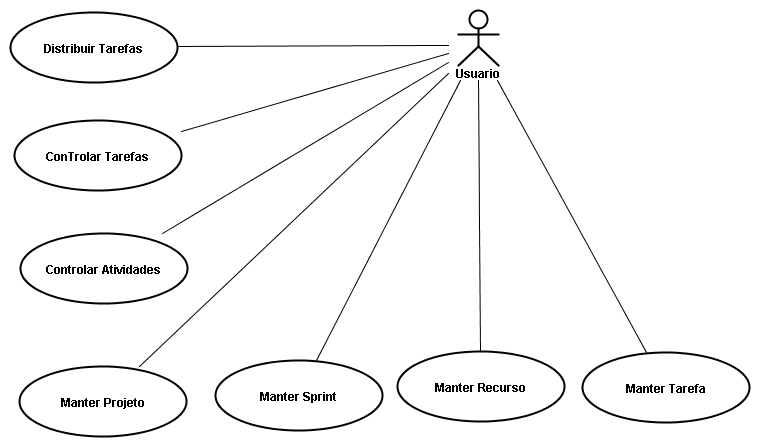


Figura 3: Diagrama de Casos de Uso do sistema

Fonte: Elaborada Pelo Autor

Figura 4: Descrição dos Casos de Uso do Sistema

|  |  |  |  |
| --- | --- | --- | --- |
| **Código** | **Nome** | **Atores** | **Descrição** |
| UC01 | Distribuir Tarefas | Usuário | Distribuir tarefas entre os recursos no Sprint |
| UC02 | Controlar Atividades | Usuário | Gerenciar as atividades de cada tarefa, com relação a sua execução. |
| UC03 | Controlar Tarefas | Usuário | Gerenciar as tarefas com relação a sua execução. |
| UC04 | Manter Projeto | Usuário | Inserir, editar, excluir e listar projetos |
| UC05 | Manter Sprint | Usuário | Inserir, editar, excluir e listar Sprints |
| UC06 | Manter Recurso | Usuário | Inserir, editar, excluir e listar Recursos |
| UC07 | Manter Tarefa | Usuário | Inserir, editar, excluir e listar Tarefas |

Fonte: Elaborada Pelo Autor

Com o diagrama de casos de uso apresentado e seguindo a análise para o desenvolvimento do sistema de distribuição de tarefas, foi gerado o diagrama de domínio que representa as entidades que fazem parte do universo de solução do sistema. Essas entidades são modeladas contento apenas os atributos.

A figura 5 mostra o diagrama de domínio modelado para o sistema de distribuição de tarefas:

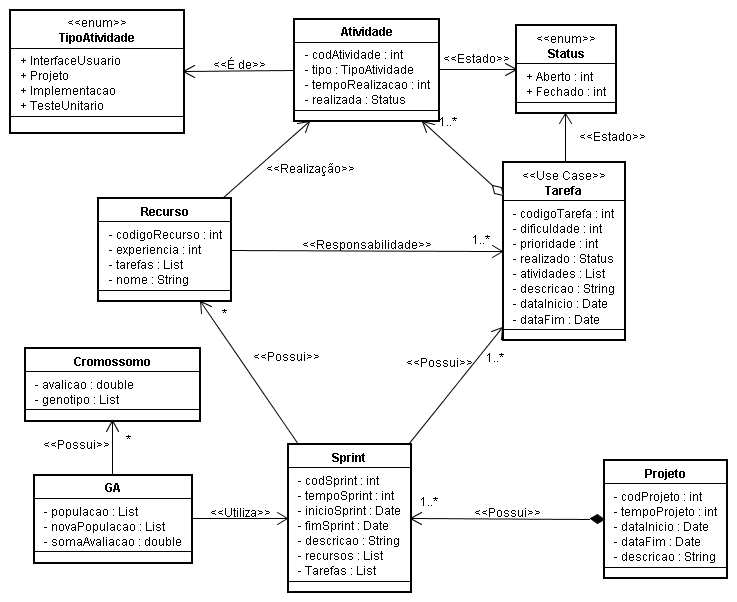


Figura 5: Diagrama de Domínio do Sistema

Fonte: Elaborada Pelo Autor

A partir do digrama de domínio foi feita toda a implantação do sistema de distribuição de tarefas.

### IMPLEMENTAÇÃO DO SISTEMA

A implementação do sistema foi divida no desenvolvimento de três módulos, no qual cada um deles atende a um requisito funcional. Os módulos implementados foram o de distribuição de tarefas, gerenciamento de tarefas e atividades e cadastros.

No presente trabalho será explanado mais detalhadamente o módulo de distribuição de tarefas, por se tratar do principal módulo do sistema, no qual foi utilizado todas as tecnologias apresentadas até aqui. Portanto, os outros módulos serão apenas apresentados.

#### **MÓDULO DE DISTRIBUIÇÃO DE TAREFAS**

O módulo de distribuição de tarefas foi implementado para suprir a necessidade mostrada pelo requisito funcional **F02**, Distribuir Tarefas, que diz que em sua descrição que o sistema deve ser capaz de distribuir tarefas para recursos levando em consideração às especificidades da metodologia de desenvolvimento ágil Scrum.

A construção desse módulo foi feita utilizando um método de mineração de dados baseado em algoritmos genéticos.

Para chegar a uma solução satisfatória com a utilização de um algoritmo genético foi feito um estudo nas entidades que compõe o problema, recursos e tarefas, para saber como poderiam ser usadas,

Esse estudo foi feito observando um ambiente de desenvolvimento, que utiliza a metodologia *Scrum.* Aqui as tarefas tinham sua representação da seguinte forma ilustrada na tabela 8:

Tabela 8: Representação de Tarefa no ambiente de desenvolvimento

|  |  |  |
| --- | --- | --- |
| **Nome** | **Prioridade** | **Dificuldade** |
| UC01 | 1 | 5 |

Fonte: Elaborada Pelo Autor

Nessa representação a prioridade representa o grau de importância que essa tarefa tem dentro do período de desenvolvimento, *Sprint*,e a Dificuldade é uma representação numérica de quanto difícil é resolver essa tarefa.

Esses valores foram estimados no inicio do *Sprint*, utilizando uma técnica de estimativa ágil, o *Plaing Poker*, no qual a equipe de desenvolvimento decide, por meio de um jogo, o quão difícil é determinada tarefa, e valoram a dificuldade a tarefa quando todos estiverem de comum acordo.

Depois de estimadas, as tarefas foram colocadas no Sprint segundo sua prioridade e foram selecionadas para o desenvolvimento de acordo com a escolha de cada membro da equipe.

No fim do *Sprint,* foi feito uma avaliação do desenvolvimento das tarefas. Foi observado que cada ponto de dificuldade, para ser resolvido, era despendido um dia de esforço, em média, pelo recurso.

Desse resultado, que diz que um ponto de dificuldade gera um esforço de um dia, foi possível fazer um paralelo entre a dificuldade da tarefa e a capacidade de resolução do recurso. Essa a capacidade foi chamada de Experiência do recurso.

A relação entre experiência do recurso, dificuldade da tarefa e esforço pode ser visto na equação abaixo:

Equação 1 : Relação entre Dificuldade e Experiência



Onde:

* **Es -**  é o esforço despendido para a resolução da tarefa
* **Di -** é a dificuldade da tarefa
* **Ex -** Experiência do recurso

Definida essa relação foi possível viabilizar a implantação do módulo de distribuição de tarefas baseado em um método de mineração de dados que utiliza como técnica de mineração o algoritmo genético.

#### **IMPLEMENTANDO O MÓDULO DE DISTRIBUIÇÃO DE TAREFAS**

Para construir o módulo de distribuição de tarefas tem que ser utilizado algoritmo genético já que esse é um requisito não funcional do sistema.

Para implementar um GA são necessários atender algumas exigências, que são a de escolher a representação do cromossomo e definir a função de avaliação.

##### **REPRESENTAÇÃO DO CROMOSSOMO**

Para o sistema de distribuição de tarefas, a representação adotada para os indivíduos foi uma representação baseada em lista de objetos.

O cromossomo é composto por uma lista objetos, onde cala índice da lista é um objeto Recurso. Cada objeto Recurso contém uma lista de objetos tarefas. A figura 6 ilustra a representação do individuo dentro do sistema:

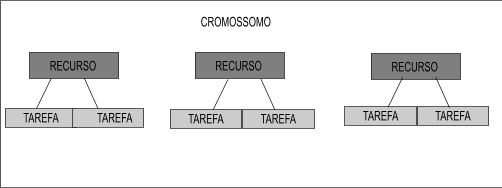


Figura 6: Representação do Cromossomo (Individuo)

Fonte: Elaborada Pelo Autor

Essa configuração mostra que cada recurso é um gene que forma o cromossomo. Para inicializar o elemento é utilizado um método randômico que relaciona recursos e tarefas que estão alocados em um determinado *Sprint*.

De posse da representação da solução, cromossomo, é possível gerar a população inicial de soluções do problema. Essa população é gerada de forma aleatória.

A listagem 1, representam os blocos de código que representam as inicializações do elemento e da população respectivamente.

**Listagem 1 Bloco de código que represneta a inicialização do elemento**

public void InicializaElemento(Sprint sprint)

{

while (sprint.ListaRecursos.Count > 0)

{

Recurso r = new Recurso();

Random rd = new Random();

double tempoExecucao = sprint.Tempo;

r = sprint.ListaRecursos[rd.Next(0, sprint.ListaRecursos.Count)];

while ((tempoExecucao > 0) &&

((this.TarefaMenorDificuldade(sprint).Dificuldade/r.Experiencia) <= tempoExecucao))

{

if (sprint.ListaTarefas.Count <= 0)

{

break;

}

Tarefa t = new Tarefa();

t = sprint.ListaTarefas[rd.Next(0, sprint.ListaTarefas.Count)];

if ((t.Dificuldade / r.Experiencia) > tempoExecucao)

{

break;

}

else

{

tempoExecucao -= t.Dificuldade / r.Experiencia;

r.Tarefas.Add(t);

sprint.ListaTarefas.Remove(t);

}

if ((tempoExecucao == this.TarefaMenorDificuldade(sprint).Dificuldade))

{

r.Tarefas.Add(this.TarefaMenorDificuldade(sprint));

sprint.ListaTarefas.Remove(this.TarefaMenorDificuldade(sprint));

break;

}

}

genotipo.Add(r);

sprint.ListaRecursos.Remove(r);

}

this.sprint = sprint;

}

**Listagem 2: Bloco de código para inicialização da população de cromossomos**

public void InicializaPopulacao()

{

while (this.somaAvalicoes < 200)

{

Inicializador i = new Inicializador();

Cromossomo c = new Cromossomo();

c.InicializaElemento(i.GetSprint());

c.SetAvaliacao();

this.somaAvalicoes = this.somaAvalicoes + c.Avliacao;

populacao.Add(c);

}

}

Um ponto importante, sobre a população inicial, é que, essa deve ter diversidade suficiente, para permitir ao algoritmo, combinar características e gerar novas soluções.

Além do individuo e da população inicial, outro requisito para a implantação de um algoritmo genético, é a função de avaliação, essa consiste, em um método para medir a qualidade do individuo, ou seja, uma forma de avaliar o quanto o individuo é apto para ser solução do problema.

Para o problema de distribuição de tarefas no *Scrum,* que é problema o ser resolvido pelo GA aqui implementado, a função de avaliação foi deduzida a partir da relação entre o esforço, experiência e dificuldade.

A lógica usada para chegar à função de avaliação foi a seguinte: levando em consideração que o *Sprint* é uma caixa de tempo que deve ter uma duração determinada, ou seja, tem um intervalo bem definido entre o seu inicio e fim.

Com isso todos os recursos alocados a um *Sprint* têm, por conseqüência, o tempo do *Sprint* para trabalharem e suas tarefas, este tempo de trabalho por tarefa é definido pelo esforço.

Também cada tarefa que está dentro do *Sprint*, tem seu tempo. Esse é definido por sua dificuldade. No ambiente observado a relação entre o tempo e a dificuldade da tarefa é de 1 para 1, ou seja, se a tarefa tem um ponto de dificuldade ele leva um dia para ser resolvida.

De posse destas informações foi deduzida a seguinte função de avaliação para a resolução do problema de distribuição de tarefas para a metodologia Scrum:

Equação 2: Função de Avaliação



Onde:

* A é a avaliação do individuo.
*  É o somatório dos esforços dos recursos do *Sprint*.
*  É o somatório das dificuldades das tarefas que estão no *Sprint.*

O trecho de código da listagem 3, demonstra como é obtida a avaliação dos cromossomos em nível de código:

**Listagem 3: Bloco de código para avaliação dos cromossomos**

public void SetAvaliacao( )

{

double TempoExecucao = 0;

for (int i = 0; i < this.genotipo.Count; i++)

{

for (int j = 0; j < this.genotipo[i].Tarefas.Count; j++)

{

TempoExecucao += this.genotipo[i].Tarefas[j].Dificuldade;

}

}

this.avliacao = TempoExecucao /sum(tarefas.dificuldade) ;

}

Uma parte importante dos algoritmos genéticos e é seleção dos pais, este procedimento serve para posteriores cruzamentos entre indivíduos para que sejam geradas soluções melhores a cada geração. O método utilizado foi um pseudo-elitismos e apenas são selecionados para se reproduzir aqueles indivíduos que tenham sua avaliação dentro de um intervalo determinado. A listagem 4, mostra o trecho de código que implementa o método de seleção dos pais.

**Listagem 4: Bloco de código que implementa o método de seleção dos pais.**

public Cromossomo Elitismo()

{

Cromossomo aux = new Cromossomo();

foreach(Cromossomo c in this.populacao)

{

if ((c.Avliacao>=0.6)&&(c.Avliacao<=1.0))

{

aux=c;

break;

}

}

return aux;

}

Depois da seleção é importante transferir as informações obtidas para novos indivíduos que serão formados a partir destas informações. Este processo e chamado de cruzamento ou recombinação.

O método de recombinação utilizado para o sistema de distribuição de tarefas escolhido foi o de *crossover* de um ponto, que consiste em determinar um ponto de corte e tomar os valores de um pai a esquerda do ponto de corte e os valores de outro pai a direita do ponto de corte e assim criar um novo individuo a partir dessa troca de informação dos pais. A listagem 5, mostra o trecho de código que implementa o método de recombinação:

**Listagem 5: bloco de código que implementa o método de recombinação**

public Cromossomo Crossover(Cromossomo c)

{

Cromossomo cromo = new Cromossomo();

Random rd = new Random();

int pontoCorte = rd.Next(0, this.genotipo.Count);

for(int i = pontoCorte; i<this.genotipo.Count-1;i++)

{

this.genotipo[i] = c.Genotipo[i];

}

cromo.Genotipo = this.genotipo;

return cromo;

}

O filho gerado pelo cruzamento será adicionado na nova população a cada iteração. A listagem 6, mostra o trecho de código para o módulo de geração:

Listagem 6: Bloco de código para o módulo de gerarção

public void Geracao()

{

novaPopulacao = new List<Cromossomo>();

Cromossomo pai1, pai2, filho;

for (int i = 0; i < this.populacao.Count; i++)

{

pai1 = this.Elitismo();

pai2 = this.Elitismo();

filho = pai1.Crossover(pai2);

novaPopulacao.Add(filho);

}

}

Outra questão importante a ser observada na implantação de uma solução utilizando GA, é a substituição da população, que por uma questão de simplicidade, se assume que, a população em questão, não pode crescer. Logo os indivíduos mais antigos são substituídos, quando os mais novos, vão nascendo. Isso acontece por considerar que o universo de solução é pequeno demais para ambos existirem.

Essa operação, é feita da maneira mais simples, apenas colocando uma nova população no lugar da antiga. A listagem 7, mostra o trecho de código abaixo mostra essa operação.

Listagem 7: Bloco de código que ilustra a operação de substituição de população

public void ModuloPopulacao()

{

if (this.novaPopulacao.Count > 0)

{

this.populacao = this.novaPopulacao;

}

}

A interface que dá acesso ao módulo de distribuição de tarefas dos sistemas, é ilustrada na figura 14:

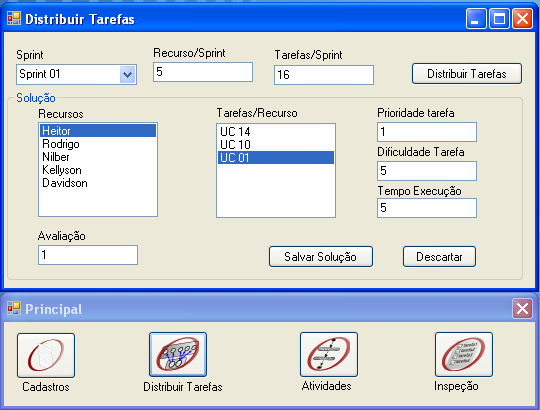


Figura 14: Tela do Módulo e Distribuição de Tarefas

Fonte: Elaborada Pelo Autor

Aqui o usuário vai apenas selecionar um *Sprint*, e clicar no botão distribuir tarefas após isso será preenchido, na área de solução, a lista com os nomes dos recursos e o valor da avaliação da solução. A avaliação apresentada é a da melhor solução dentro da população que foi gerada a partir do GA. Os valores para a avaliação dos indivíduos variam no intervalo de 0 a 1.

Ao clicar em um elemento que está na lista de recursos é preenchida a lista de tarefas com aquelas que são de responsabilidade do recurso selecionado. Clicando em uma tarefa é exibida a sua prioridade dentro do *Sprint*, sua dificuldade e seu tempo de execução (Esforço).

Se a solução apresentada é satisfatória clica-se no botão salvar solução esta será gravada para o desenvolvimento do *Sprint*, se não basta clicar em descartar e gerar uma nova solução.

#### **MÓDULO DE CADASTROS**

O módulo de cadastro do sistema de distribuição de tarefas é responsável por toda a parte de manutenção dos dados do projeto que está sem desenvolvido e os dados do processo Scrum, *Sprint*, recursos e tarefas.

A parte de manutenção serve para as operações de inserção, edição, deleção e listagem de projeto, *Sprint*, recursos e tarefas.

A interface é composta por um conjunto de abas onde cada aba é responsável por uma entidade. Na aba se encontra um uma seleção para entidade, um botão de editar e deletar, os campos que compõem a entidade e os botões de salvar e limpar.

O processo para a manutenção é o seguinte, se a operação a ser feita e a de inserção (cadastro), basta preencher os campos e clicar no botão salvar, para a edição, primeiro e necessário fazer a seleção da entidade e clicar no botão Editar, os campos serão preenchidos com os valores, fazer a alteração do valor no(s) campo(s) e clicar em salvar. Para fazer a deleção basta selecionar a entidade e clicar no botão deletar. Abaixo as figuras de cada aba do módulo de cadastro.

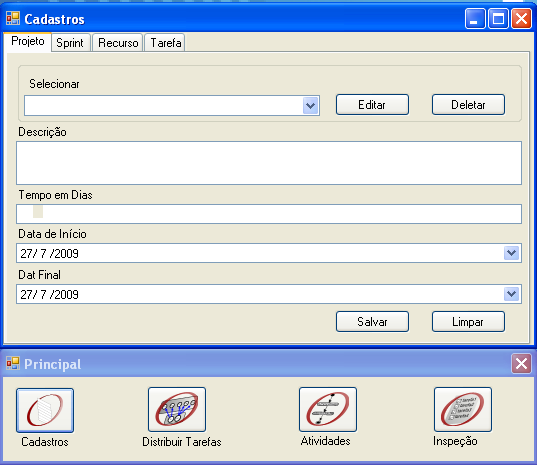


Figura 15: Cadastro De Projetos

Fonte: Elaborada Pelo Autor

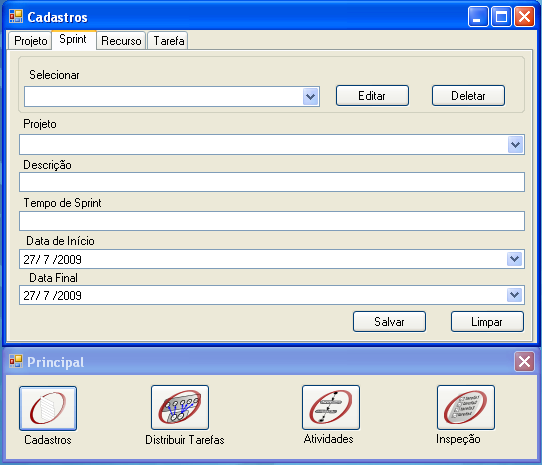


Figura 16: Cadastro de Sprint

Fonte: Elaborada Pelo Autor

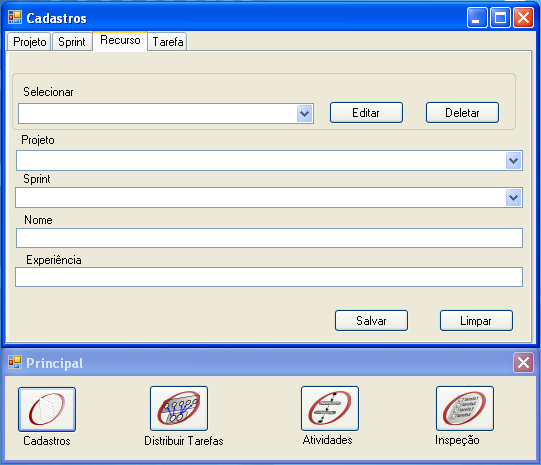


Figura 17: Cadastro de Recursos

Fonte: Elaborada Pelo Autor

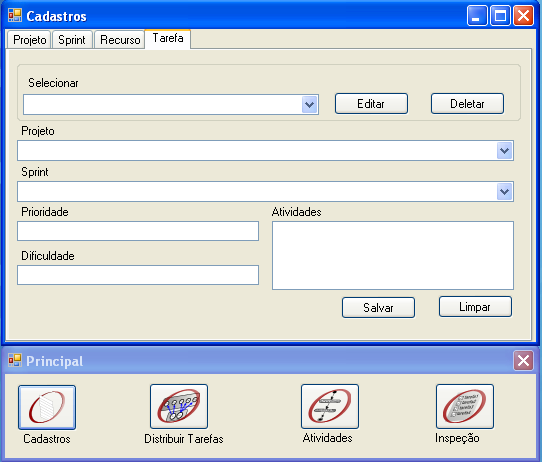


Figura 18: Cadastro de Tarefas

Fonte: Elaborada Pelo Autor

#### **MÓDULO DE GERENCIAMENTO DE TAREFAS**

O módulo de gerenciamento de tarefas, serve para avaliar o andamento do *Sprint,* ou seja, como recursos estão desenvolvendo as atividades e por conseqüência suas tarefas, já que cada tarefa tem um conjunto de atividades para serem realizadas para que a tarefa seja realizada.

Este módulo e formado por duas interfaces, uma para o gerenciamento de atividades e outra para o gerenciamento de tarefas.

Na interface de gerenciamento de atividades escolhe-se o recurso e a tarefa. Depois da escolha da tarefa é preenchida a lista de atividades que fazem parte daquela tarefa.

As informações que serão acrescentadas para o gerenciamento da atividade é o tempo gasto para o término da atividade e o status da atividade.

A figura abaixo mostra a interface para o gerenciamento das atividades de cada tarefa:

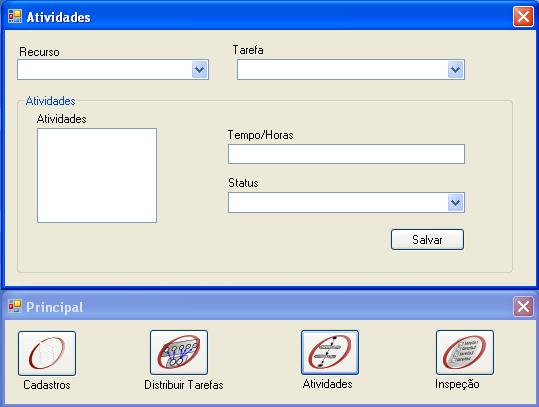


Figura 19: Gerenciamento de Atividades

Fonte: Elaborada Pelo Autor

Na parte de gerenciamento de tarefas é apenas uma inspeção de que tarefa determinado recurso terminou. A figura abaixo mostra o módulo de gerenciamento de tarefas:



Figura 20: Gerenciamento de Tarefas

Fonte: Elaborada Pelo Autor

Esse módulo é basicamente para avaliar o desempenho da equipe no decorrer do *Sprint.* E avaliar a estimativa das dificuldades das tarefas para ver se foi sub ou superestimadas, baseadas nas datas de fechamento de cada uma delas.

O fechamento da tarefa, status, é automaticamente feito quando todas as atividades que compõem aquela tarefa estiverem fechadas.

# Considerações Finais

Neste trabalho foi mostrado o desenvolvimento de um sistema de distribuição de tarefas para a metodologia *Scrum.* Esse deveria utilizar um método de mineração de dados baseado em algoritmos genéticos.

Além da distribuição de tarefas o sistema também gerencia as tarefas, ou seja, da a condição do usuário visualizar as tarefas que foram terminadas, o tempo que levou para serem resolvidas e avaliar tanto o desempenho do recurso como a estimativa da tarefa.

O sistema foi desenvolvido para resolver um problema similar ao de escalonamento de tarefas, que consiste em alocar tarefas, que para serem resolvidas, precisam passar por um conjunto de máquinas, o que foi feito para o desenvolvimento do sistema foi uma adaptação desse problema para um ambiente de desenvolvimento de software utiliza a metodologia ágil Scrum.

O resultado esperado com o sistema era que, de acordo com os componentes de um *Sprint*, ou seja, os recursos e as tarefas naquele locado, fosse feita uma distribuição onde todos os recursos tivessem tarefas para resolver e tivessem tempo hábil para resolvê-las.

O sistema mostrou um comportamento satisfatório com relação a distribuição de tarefas, sempre atendo as expectativas, de todos os recursos terem tarefas, e aqueles terem tempo necessário para resolvê-las. Com o sistema foi possível observar os seguintes fatos:

* Existe, uma convergência ao resultado de melhor avaliação gerado pelo GA, quando, o somatório das dificuldades das tarefas é menor do que, o somatório do tempo *Sprint* dos recursos, vezes o somatório de experiência dos recursos;
* No entanto, quando o somatório das dificuldades de todas as tarefas dentro do *Sprint* supera o de experiência, a distribuição nunca chega a utilizar todas as tarefas na distribuição, essa característica é boa de ser observada, no começo do *Sprint*, onde a equipe, pode retirar tarefas para equilibrar o valor de dificuldade com o de experiência dentro do *Sprint.*
* O sistema não faz um equilíbrio do tempo ocioso dos recursos, ou seja, se houver de alguma forma, um tempo ociosos, por causa da relação de Di /Ex, o sistema não dividirá esse tempo igualmente entre os recursos, podendo um recurso ter todo o seu Sprint preenchido com tarefas, e outro não.
* Outra vantagem que se tem, é a de toda equipe, independente da experiência de cada recurso, poder resolver tarefas de dificuldades variadas, sem o direcionamento natural de menos experiência para dificuldades menores, acarretando uma estagnação da experiência do recurso.
* Além de distribuir as tarefas, o sistema também dá suporte a gerência de tarefas, ou seja, dar a equipe a visão geral do desenvolvimento do Sprint, através do módulo de gerência. Este módulo, dar a visão das atividades sendo resolvidas e, por conseqüência, as tarefas, dando a possibilidade de avaliar o desenvolvimento dos recursos e, as estimativas das tarefas.

Com o exposto acima, se observou que os algoritmos genéticos tiveram um bom desempenho para atacar o problema, para o qual, o sistema apresentado no decorrer deste trabalho, se destina a solucionar.

O sistema precisa de mais testes em ambientes reais de desenvolvimento, para mostrar sua real contribuição para a equipe, mas em testes controlados se mostrou satisfatório e coerente nos resultados obtidos.

## **TRABALHOS FUTUROS**

Neste trabalho foi desenvolvido um sistema de distribuição de tarefas para um projeto de desenvolvimento de software no qual o processo de desenvolvimento utilizado é o Scrum.

Para a solução do problema foi utilizado um algoritmo genético de simples funcionamento e implantação que apenas gera soluções satisfatórias.

Para a melhoria do sistema pode se programar um GA mais robusto, com os métodos genéticos mais adaptados ao problema, com uma população que não seja substituída a cada geração.

Além das melhorias no GA também seria interessante a utilização de uma tecnologia de mineração hibrida com Redes neurais, para que o sistema seja capaz de, a partir da configuração do *Sprint*, ou seja, recursos e tarefas, seja capaz de sugerir uma solução para essa configuração.

Pode também ser feito, para a melhoria do sistema, uma adaptação para que ele seja genérico com relação aos processos, isto é, ser capaz de distribuir tarefas para qualquer tipo de processo o qual seja instanciado dentro do sistema.

Essas melhorias expostas acima podem ser atacas em trabalhos futuros relacionados a esse.

# REFERÊNCIAS BIBLIOGRÁFICAS

AMBLER, Scott W. **Modelagem Ágil: Práticas Eficazes para a Programação eXtrema e o Processo Unificado**. Tradução: Acauan Fernandes. Porto Alegre: Bookman, 2004.

SOMMERVILLE, Ian. **Engenharia de *Software***. Tradução: Selma Shin Shimizu Melnikoff, Reginaldo Arakaki, Edilson de Andrade Barbosa. 8 ed. São Paulo: Pearson, 2007.

KOCH, ALAM. S. **Agile Software Development Evaluating the Methods for Your Organization**. 1 ed. ARTECH HOUSE, 2005.

SCHWABER, KEN. **Agile Project Management with Scrum**. 1 ed. Microsoft Press, 2004.

LINDEN, RICARDO, **Algoritmos Genéticos**. 1ed Rio de Janeiro: Brasport, 2006.

GOLDSCHMIDT,RONALDO;PASSOS,EMANUEL, **Data mining um guia prático**. 3 ed. Rio de Janeiro: Elsivier, 2005.

VALENTIM, ESDRAS. S. **Estado da Arte em uma análise da aplicação dos algoritmos genéticos para determinação de roteiros turísticos**. IFRN. Natal. 2008

PEREIRA, PAULO; TORREÃO, PAULA; MARÇAL, ANA.S. ***Entendendo Scrum para gerenciar processo de forma ágil***. Scrum, 2007. Disponível em: < http://www.cesar.org.br/files/file/SCRUM\_MundoPM-Abril-Maio-2007.pdf>. Último Acesso em maio de 2009.

SCHWABER, KEN. **SCRUM Development Process**. Scrum, 2004.< <http://home.hib.no/ai/data/master/mod251/2009/articles/scrum.pdf>>. Último Acesso em maio de 2009.