PRODUÇÃO ONLINE
REVISTA CIENTÍFICA ELETRÔNICA DE ENGENHARIA DE PRODUÇÃO
ISSN 1676-1901

MINIMIZAÇÃO DOS TEMPOS DE ATRASO NA PROGRAMAÇÃO DE TAREFAS EM UMA EMPRESA DE DESENVOLVIMENTO DE SOFTWARES

MINIMIZING TOTAL TARDINESS IN A SOFTWARE DEVELOPING COMPANY

Icaro Paulo Ludwig* - E-mail: icaroludwig@gmail.com
Michel José Anzanello* - E-mail: anzanello@producao.ufrgs.br
Gabriel Vidor* - E-mail: gvidor@producao.ufrgs.br
*Universidade Federal do Rio Grande do Sul, Porto Alegre, RS

Resumo: Empresas de pequeno porte do setor de serviços, como as desenvolvedoras de softwares, usualmente programam suas tarefas de forma manual. Tal programação é viável enquanto o número de tarefas é reduzido, mas acarreta dificuldades gerenciais à medida que crescem, implicando em atrasos na entrega de tarefas. Este artigo tem como objetivo utilizar uma ferramenta de sequenciamento como forma de minimizar esses atrasos. Para isso, propõe duas heurísticas para a programação de tarefas embasadas nos seguintes passos: (*i*) definição de um ordenamento inicial para as tarefas; (*ii*) alocação das tarefas aos times de desenvolvimento; e (*iii*) ordenação das tarefas em cada time de desenvolvimento com vistas à minimização do atraso total. As heurísticas propostas reduziram os atrasos de entregas em dados reais e simulados, simplificaram o processo de programação e possibilitaram maior visibilidade do processo de desenvolvimento.

Palavras-chave: Sequenciamento. Atraso. Software. Gestão. Desenvolvimento.

Abstract: Small companies in service sectors, such as software developers, usually rely on manual-based programming tasks. That programming yields satisfactory results for small task lists, but leads to managerial difficulties as a large number of tasks increases task delays. This paper aims at using scheduling tools to minimize such delays. For that it proposes two heuristics for task scheduling based on the following steps: (i) define an initial order for tasks, (ii) distribute each task to development teams, and (iii) schedule the tasks in each development team aimed at minimizing total tardiness. The proposed approach reduced the total tardiness in simulated in real data, simplified the process of scheduling and provided better tracking of the development process.

Keywords: Scheduling. Tardiness. Software. Management. Development.

1 INTRODUÇÃO

Desde a difusão do Sistema Toyota de Produção (STP) arquitetado por Ohno (1997), a correta previsão de prazos de entrega tem sido um requisito básico nas relações entre cliente e fornecedor. Embora a grande maioria das estratégias de produção tenha se originado na indústria, elas têm sido absorvidas por diferentes áreas, como no setor de serviços (CANEL et al., 2000). Neste contexto, empresas

desenvolvedoras de software aparecem como exemplos relevantes de prestadoras de serviços.

O setor de Tecnologia da Informação (TI), onde as empresas de desenvolvimento de software estão inseridas, tem cumprido apenas 34% de seus projetos dentro do planejamento inicial, atendendo prazo, custo e funcionalidades (THE STANDISH GROUP, 2009). Tal situação demanda o desenvolvimento de ferramentas robustas para assegurar o cumprimento dos prazos de entrega dos produtos aos clientes (LAW e IP, 2011).

Empresas que trabalham com projetos de software apresentam etapas que se assemelham a uma indústria de bens, por vezes até se denominando “Fábrica de Softwares” (FERNANDES e TEIXEIRA, 2007), sob a filosofia de Engenharia sob Encomenda – *Engineer to Order* (FRAMINAN e RUIZ, 2010).

Por conta do elevado nível de customização do setor de software, parte significativa das tarefas pode estar sendo executada pela primeira vez. Isso faz com que não exista uma medição prévia de tempo e esforço necessário para sua conclusão. As estimativas são feitas por similaridade com tarefas que possuam características conhecidas, introduzindo assim incerteza às mesmas.

Além do planejamento inicial, também se fazem necessárias constantes revisões de *status* no andamento das tarefas, ratificando as estimativas iniciais de duração e permitindo correções imediatas em caso de desvios (SZÖKE, 2011). Somam-se a isso outras variáveis que desafiam a programação no desenvolvimento de software: a adição de tarefas extras ao longo do projeto, o compartilhamento de recursos entre diferentes projetos, e a dependência do fornecimento de dados e da disponibilidade de agenda do próprio cliente (APEL e KASTNER, 2009).

Desta forma, sistemas informatizados de Planejamento, Programação e Controle da Produção (PPCP) podem trazer agilidade na identificação de desvios do planejamento em ambientes expostos às interferências anteriormente citadas (WANG e WANG, 2012).

Este trabalho propõe duas heurísticas de sequenciamento com vistas à minimização do atraso total de entrega de tarefas em uma empresa de desenvolvimento de softwares personalizados. As heurísticas são compostas por três passos operacionais: (*i*) ordenamento inicial das tarefas para subsequente alocação, (*ii*) distribuição das tarefas às equipes de desenvolvedores, e (*iii*)

ordenamento das tarefas alocadas a cada equipe com vistas à minimização do atraso total. As heurísticas são inicialmente testadas em um banco de dados simulados e então aplicadas a dados reais.

A principal contribuição do artigo está na proposição de heurísticas de sequenciamento que ofereçam suporte na aceitação de pedidos e maior controle no progresso da execução dos mesmos. Isso faz com que a programação de tarefas, tradicionalmente embasada em procedimentos empíricos executados por membros da gerência, seja eficientemente substituída por procedimentos computacionais de simples implementação. A programação de tarefas passa então a ser executada pelo pessoal de apoio da empresa, liberando gerentes para outras atividades.

O presente artigo apresenta, na seção 2, os fundamentos da programação de tarefas (sequenciamento), bem como uma breve descrição do setor de software. A seção 3 detalha os passos do método proposto, enquanto que a seção 4 traz os resultados e as discussões do trabalho. Na seção 5 são apresentadas as conclusões do estudo.

2 REFERENCIAL TEÓRICO

2.1 O Setor de Software

Segundo a Associação Brasileira das Empresas de Software, o setor teve faturamento aproximado de US$ 885 bilhões no mundo durante o ano de 2010. Deste montante, US$ 19 bilhões foram movimentados no Brasil, indicando um crescimento de 24% em relação ao ano anterior. Dentre as 8,5 mil empresas sediadas no país, cerca de 94% são classificadas como micro ou pequenas empresas, e 76,5% trabalham no desenvolvimento, distribuição e comercialização de softwares; as demais se dedicam a serviços relacionados (ABES, 2010).

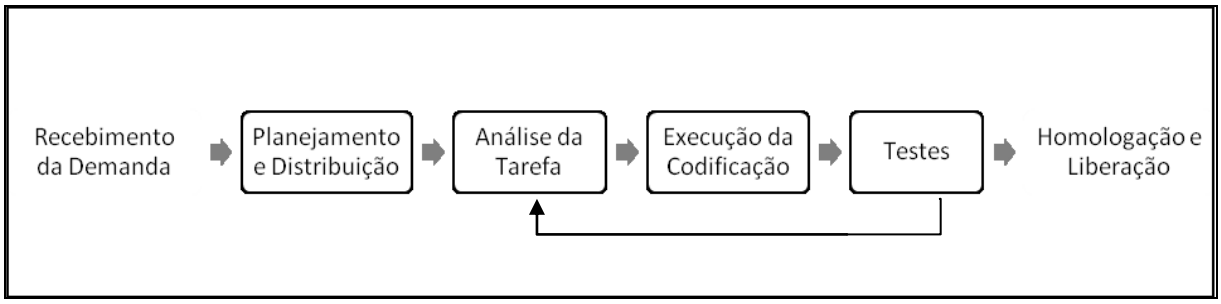
O processo de criação e desenvolvimento de software se concentra em duas grandes frentes: (*i*) o desenvolvimento de produtos, e (*ii*) o desenvolvimento de projetos personalizados. Em diversos casos também se observa um combinado entre ambos: um produto, com características padrão em sua essência, é personalizado para atender a funcionalidades específicas. Cada processo possui estratégias de atuação e formas de gerenciamento específicas. Enquanto o primeiro

grupo adota métodos e ferramentas de desenvolvimento de produto (com forte interação com o mercado durante a concepção da ideia e posterior condução do projeto internamente), o segundo grupo exige alto grau de interação com o cliente, mesmo durante a fase de construção e programação, o que implica em um cenário mais dinâmico (FERNANDES e TEIXEIRA, 2007).

Em ambos os casos existem dificuldades comuns, como a forma de levantamento dos requisitos que devem compor o escopo e a forma de estimar acertadamente esforço, prazos e custos. Acrescenta-se a isso o fato do desenvolvimento ser realizado por recursos humanos caracterizados por elevada variabilidade, o que, por vezes, inviabiliza a estimativa adequada de prazos de entrega (THE STANDISH GROUP, 2009).

As principais referências em gerência de projetos para o setor de TI, como o PMI (2004) e o SEI (2006), não impõem o uso de uma ferramenta específica para organizar o andamento e monitorar a execução dos projetos, ficando esse processo a critério da empresa. Para Fernandes e Teixeira (2007), diversas atividades do processo de desenvolvimento de software - como o desenvolvimento de códigos - assemelham-se a uma linha de montagem industrial. O uso de ferramentas administrativas, gerenciais e de controle operacional, tipicamente associado a ambientes industriais, emerge como útil neste tipo de empresa. A Figura 1 traz um exemplo de processo de codificação de programas usualmente utilizado em uma fábrica de software.

Figura 1 – Modelo de Fábrica de Software.



Fonte: Fernandes e Teixeira (2007)

Por conta da semelhança com linhas de montagem, verifica-se a possibilidade de implementação de ferramentas originalmente introduzidas pela Engenharia de Produção para a indústria em processos de desenvolvimento de

software. Tais ferramentas incluem métodos de programação de tarefas, tradicionalmente conhecidas como sequenciamento.

2.2 Programação da Produção e Sequenciamento

Tubino (2007) faz uma divisão conforme os horizontes de tempo para o Planejamento da Produção: (*i*) longo prazo, o qual chama de Planejamento Estratégico da Produção; (*ii*) médio prazo, onde se encontra o Plano-mestre da Produção, que atua de forma tática; e (*iii*) curto Prazo, onde está a Programação da Produção, caracterizada por cunho operacional.

No escopo operacional, Davis et al. (2001) apontam os objetivos mais comuns da Programação da Produção: atender às datas de entrega dos clientes; minimizar o tempo total de fluxo ou de processamento; minimizar estoques em processo; minimizar o tempo ocioso das máquinas e dos trabalhadores; minimizar os tempos de atravessamento ou *leadtimes*; e minimizar os tempos e/ou custos de *setup*.

Segundo Pinedo (2008), uma das ferramentas usadas para se atingir os objetivos da Programação da Produção é o sequenciamento, o qual é descrito como um processo decisório que lida com a alocação de recursos às tarefas, em períodos determinados de tempo, com a finalidade de otimizar um ou mais objetivos.

Alinhados aos objetivos da programação de tarefas, encontram-se as chamadas funções objetivo do sequenciamento (PINEDO, 2008): (*i*) *makespan*: equivalente ao tempo necessário até a conclusão de última tarefa. Minimizar o *makespan* geralmente implica em boa utilização dos recursos produtivos, tipicamente máquinas ou trabalhadores; (*ii*) tempo total de fluxo: através do tempo necessário à conclusão das tarefas pode-se ter ideia dos estoques intermediários e custos associados ao mesmo; (*iii*) atraso máximo: maior tempo de atraso dentre as tarefas programadas em relação à data devida; (*iv*) tarefas com atraso: informa o número de tarefas em atraso; (*v*) tempo de atraso total: soma dos tempos de atraso de todas as tarefas; e (*vi*) tempo de antecipação total: inverso do tempo de atraso total, sendo soma das diferenças dos tempos entre a data de conclusão e a data prometida das tarefas, quando estas são concluídas antes do prazo. A antecipação da conclusão das tarefas pode incorrer em custos de estoque.

Para o atendimento dos objetivos, o sequenciamento faz uso de Regras de Priorização. Tais regras podem ser heurísticas que, a partir de informações sobre as tarefas e condições do sistema produtivo, selecionam qual dentre as tarefas na fila de um grupo de recursos terá prioridade de processamento (MOR e MOSHEIOV, 2012), bem como qual recurso será encarregado de sua execução. Tubino (2007) elenca algumas das regras mais comuns (Quadro 1), ressaltando que nenhuma delas é eficiente para atender a todas as situações.

Quadro 1 – Regras de Priorização mais utilizadas

Sigla	Especificação	Definição
PEPS	Primeira que entra primeira que sai	Os lotes serão processados de acordo com sua chegada no recurso.
MTP	Menor tempo de processamento	Os lotes serão processados de acordo com os menores tempos de processamento no recurso.
MDE	Menor data de entrega	Os lotes serão processados de acordo com as menores datas de entrega.
IPI	Índice de prioridade	Os lotes serão processados de acordo com o valor da prioridade atribuída ao cliente ou ao produto.
ICR	Índice crítico	Os lotes serão processados de acordo com o menor valor de: $\frac{\text{(data de entrega – data atual)}}{\text{Tempo de processamento}}$
IFO	Índice de folga	Os lotes serão processados de acordo com o menor valor de: $\frac{\text{(data de entrega – Σ tempo de processamento restante)}}{\text{Número de operações restantes}}$
IFA	Índice de falta	Os lotes serão processados de acordo com o menor valor de: quantidade em estoque / quantidade de demanda

Fonte: Tubino (2007)

As heurísticas de sequenciamento são normalmente associadas à disposição dos recursos produtivos. Dentre tais disposições, é comum encontrar grupos de recursos formados por máquinas operando em paralelo (PINEDO, 2008). Máquinas em paralelo podem apresentar capacidades ou tempos de preparação distintos, fazendo com que as ferramentas de sequenciamento tratem cada recurso de forma própria (FANDEL e HEGENE, 2006).

Tais arranjos são chamados de máquinas paralelas não-relacionadas, e figuram entre os mais complexos problemas de sequenciamento. Atividades que dependem de recursos humanos são exemplos destes arranjos, pois envolvem diferenciação de cada operador ou grupo de operadores, tanto na capacidade produtiva quanto na curva de aprendizagem (ANZANELLO e FOGLIATTO, 2010).

Pinedo (2008) sugere um algoritmo para o sequenciamento em máquinas paralelas não relacionadas apoiado em dois passos. Primeiramente, determinam-se quais tarefas serão alocadas para cada recurso; na sequência, determina-se o ordenamento das tarefas em cada um desses recursos.

O estudo de Anzanello e Fogliatto (2010), por sua vez, é focado em trabalhadores, e acrescenta um passo previamente à distribuição das tarefas aos recursos (aqui denominados times de trabalhadores). O processo de sequenciamento é dividido em três estágios: (*i*) definição da ordem inicial de distribuição das tarefas, (*ii*) alocação das tarefas aos times de forma balanceada, e (*iii*) sequenciamento do subconjunto de tarefas atribuídas para cada time de forma a atender a função objetivo.

A função objetivo tratada neste estudo - minimização do atraso total – requer elevado esforço computacional, evidenciando o *trade-off* entre a qualidade da solução obtida e a demanda de processamento computacional (LIU et al., 2006). Os métodos mais simples, como os que aplicam Regras de Priorização tradicionais, são de rápida execução, mas podem incorrer em resultados insatisfatórios, sendo denominados "*Quick and Dirty"* por Huegler e Vasko (1997).

A grande maioria das abordagens para minimização do atraso total requer elevado esforço computacional, visto que o problema é classificado como *NP-hard* (PINEDO, 2008), ou seja, a complexidade da solução tem um crescimento dito não polinomial de acordo com o crescimento do número de tarefas a serem sequenciadas.

Para a resolução de problemas de minimização do atraso total, Huegler e Vasko (1997) sugerem o uso de técnicas baseadas em Programação Dinâmica. Tais técnicas reduzem o esforço computacional, se comparadas aos métodos tradicionais como os da família de algoritmos da Programação Linear Inteira do tipo *Branch and Bound*, e encontram soluções superiores às encontradas pelas Regras de Priorização. Tais técnicas serão discutidas no método apresentado na próxima seção.

3 MÉTODO

De acordo com Yin (2003), a pesquisa aqui relatada pode ser classificada como de natureza aplicada, embasada em uma abordagem quantitativa e com objetivo exploratório. Para tanto, realizou-se um estudo de caso da aplicabilidade da ferramenta de sequenciamento em uma empresa de desenvolvimento de software, pertencente ao setor de serviços.

Pelas características do processo de desenvolvimento de software, pode-se classificar o problema como um arranjo de máquinas paralelas não-relacionadas. As heurísticas a serem aplicadas seguem o modelo de três estágios sugerido por Anzanello e Fogliatto (2010), que (*i*) define um ordenamento inicial para as tarefas a serem distribuídas; (*ii*) aloca cada tarefa aos times de desenvolvimento; e (*iii*) ordena as tarefas em cada time de desenvolvimento (cada time de desenvolvimento equivale a uma máquina). Estes passos são detalhados na sequência.

3.1 Primeiro passo: definir o ordenamento inicial para distribuição das tarefas

As tarefas de uma lista de execução são ordenadas de acordo com uma regra de priorização, formando uma fila única para posterior alocação. Neste passo duas regras são testadas:

(*i*) Tempo de execução mais curto (STP – *Shortest Processing Time*) - Esta regra prioriza a alocação das tarefas de menor duração. Tal regra não modifica o tempo total de atravessamento das tarefas, porém faz com que tarefas de rápida execução sejam prontamente liberadas para posterior processamento ou encaminhamento ao cliente (PINEDO, 2008).

(*ii*) Menor folga - A folga é a diferença entre a data de entrega da tarefa *j*, d_j, e o tempo de processamento da tarefa *j*. Esta regra prioriza a distribuição das tarefas com reduzida folga, tentando minimizar o atraso das mesmas.

3.2 Segundo passo: alocar as tarefas aos *m* times de desenvolvimento

Para a alocação das tarefas utiliza-se a regra do tempo de processamento acumulado, cujo objetivo é balancear a carga de trabalho entre os *m* times. Desta

forma, cada tarefa *j* é alocada ao time de desenvolvimento que apresentar o menor tempo acumulado de processamento até o momento. Esta regra cumpre importante função na minimização da ociosidade entre os times (ANZANELLO e FOGLIATTO, 2010).

3.3 Terceiro passo: sequenciar as tarefas alocadas a cada time de desenvolvimento

Neste passo utiliza-se o Algoritmo de Minimização do Atraso Total (AMAT), proposto por Huegler e Vasko (1997), para sequenciar as tarefas alocadas a cada time. Os *m* times de desenvolvimento dão origem a *m* problemas do tipo $1 \mid\mid \sum T_j$ (ou seja, *m* sistemas compostos por 1 time de processamento com vistas à minimização do somatório do atraso T_j dentro de cada sistema).

Esta simplificação visa viabilizar a utilização do AMAT, desenvolvido para sistemas compostos por um único time de desenvolvimento (ou máquina, no contexto tradicional de sequenciamento). Tal algoritmo foi selecionado por conta de sua relevância em aplicações práticas, robustez e reduzido esforço computacional necessário para alcançar uma solução satisfatória (HUEGLER e VASKO, 1997). Outra vantagem é que pode ser utilizado em cenários onde as tarefas apresentam datas de entregas não-comuns (ou seja, $d_i \neq d_j$).

O algoritmo AMAT decompõe o problema em etapas, que são o resultado da divisão do conjunto original de tarefas em subconjuntos menores, dividindo assim um problema com um grande número de tarefas em vários problemas menores. Na sequência é iniciado um procedimento iterativo, onde a cada iteração apenas uma etapa é sequenciada, calculando o melhor valor para a função objetivo $\sum T_j$, e respeitando os resultados parciais encontrados nas iterações anteriores. Ao final do procedimento, uma solução recomendada é obtida.

Na aplicação do AMAT, a variável *N* representa o número total de tarefas do problema, *S* o tamanho de cada subconjunto, e *PS* o ponteiro que indicará a tarefa de início de cada subconjunto. O tamanho de subconjunto *S* corresponde ao número máximo de tarefas que podem ser sequenciadas em um tempo computacional admissível (a ser definido pelo usuário). No exemplo da Figura 2, arbitrou-se *S*=8

tarefas, com tempo de processamento aproximado em 1 segundo. As etapas para aplicação do AMAT de Huegler e Vasko (1997) são agora descritas.

Etapa 1: Utilizar como sequência inicial de tarefas o resultado do ordenamento por qualquer regra de priorização. Na Figura 2, a sequência inicial é identificada pela coluna denominada Conjunto Inicial de Tarefas, a qual contém as tarefas {5,14,9,2,6,3,13,7,4,12,1,11,10,8,15}. Nesta etapa também é definido o valor inicial de *PS*, que deve apontar para a primeira tarefa de toda a lista, ou seja, a tarefa que estiver na posição 1.

Etapa 2: Formar um subconjunto de tarefas, iniciando na posição *PS* e contendo as próximas *S* tarefas; sequenciar apenas este subconjunto. Caso o subconjunto atual não inicie na primeira tarefa da lista (*PS*=1), deve-se então utilizar o devido deslocamento de tempo agregado pelas tarefas anteriores a esse subconjunto. Na Figura 2, o subconjunto formado inicialmente é destacado por um contorno tracejado. Ao final da primeira iteração, a sequência das tarefas passou a ser {7,5,13,14,9,6,2,3}, sendo apresentada na coluna “Resultado após Iteração 1”.

Figura 2 – Exemplo de aplicação do algoritmo AMAT

	Conjunto Inicial de Tarefas	Resultado após Iteração 1	Resultado após Iteração 2	Resultado após Iteração 3	Resultado após Iteração 4	Resultado após Iteração 5
	5	7	7	7	7	7
	14	5	5	5	5	5
	9	13	13	13	13	13
	2	14	14	14	14	14
	6	9	6	6	6	6
	3	6	2	2	2	2
	13	2	3	3	3	3
	7	3	12	12	12	12
	4	4	9	9	9	9
	12	12	4	4	4	4
	1	1	1	1	1	1
	11	11	11	11	11	11
	10	10	10	10	10	10
	8	8	8	8	8	8
	15	15	15	15	15	15
Atraso Total:	91	84	79	79	79	79
Aprimoramento:	-	7,7%	13,2%	13,2%	13,2%	13,2%

Fonte: Huegler e Vasko (1997)

Etapa 3: Aumentar *PS* em [*S*/2], ou seja, o próximo subconjunto a ser sequenciado iniciará na segunda metade das tarefas do subconjunto anterior. Para completar a formação do subconjunto, utilizar as *S* tarefas seguintes. Caso o final deste novo subconjunto ultrapassar o final de todas as tarefas, utilizar as últimas *S* tarefas.

Etapa 4: Repetir as etapas 2 e 3 até que todas as tarefas sejam sequenciadas. Após concluir tal sequenciamento, executar o ciclo em ordem inversa, formando inicialmente o subconjunto com as últimas *S* tarefas da lista e, em seguida, deslocando a formação dos subconjuntos em direção à primeira tarefa, conforme ilustrado nas iterações 3 a 5 da Figura 2.

Etapa 5: Executar ciclicamente as etapas 2, 3 e 4 até que nenhum aprimoramento adicional seja verificado, ou até atingir o número máximo de ciclos definido como critério de parada. Este critério de parada costuma ter como base a comparação do tempo de execução do algoritmo com o tempo disponível para a realização da programação da produção.

O Quadro 2 apresenta um resumo das duas heurísticas testadas, com vistas à minimização do atraso total.

Quadro 2 – Resumo das heurísticas utilizadas no sequenciamento

Heurística	Passo 1: Ordenamento inicial das tarefas	Passo 2: Alocação das tarefas aos times de desenvolvimento	Passo 3: Sequenciamento das tarefas em cada time
H_1	Tempo de execução mais curto	Tempo de processamento acumulado	Minimização do atraso total através da heurística baseada em programação dinâmica de Huegler e Vasko (1997)
H_2	Menor folga		

Fonte: elaborado pelos autores

Para avaliação da eficácia das heurísticas propostas, tarefas semelhantes às verificadas na empresa foram simuladas, e as heurísticas H_1 e H_2 utilizadas para sequenciar tais tarefas. As heurísticas foram então aplicadas no quadro de tarefas reais da empresa e os resultados comparados aos gerados pelo sequenciamento manual, forma atualmente empregada na empresa.

4 RESULTADOS E DISCUSSÃO

4.1 Ambiente de Aplicação

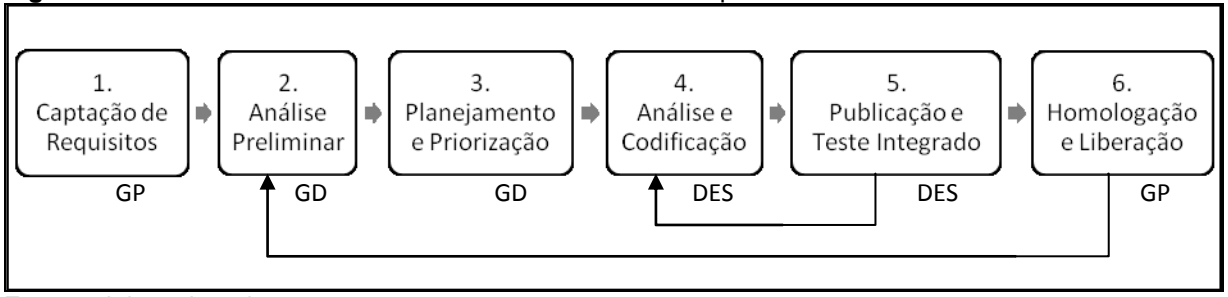
O processo de desenvolvimento de software na empresa tem suas macroatividades apresentadas no fluxograma da Figura 3. A atividade 1 corresponde à captação dos requisitos do software a ser desenvolvido, e é realizada pelos gerentes de projetos (GP). Na sequência é feita a análise preliminar do requisito pelos gerentes de desenvolvimento (GD), que verificam viabilidade técnica, pré-requisitos, tamanho de cada item e agregam detalhes adicionais ao desenvolvimento.

Na atividade 3, ordenam-se os requisitos conforme parâmetros de priorização (prazo, importância, tamanho, etc), sendo então programados para desenvolvimento. Na atividade 4, os desenvolvedores (DES) efetivamente convertem os requisitos em software, conforme a agenda prevista para a semana e respectivas priorizações. É importante enfatizar que existem diversas equipes de desenvolvedores, em paralelo, executando as tarefas da atividade 4.

Já a atividade 5 compreende a publicação dos requisitos desenvolvidos em um ambiente de homologação, cujo objetivo é atestar o correto funcionamento do requisito desenvolvido em conjunto com o restante da solução. Havendo qualquer problema nesta atividade, o processo retorna à etapa anterior para execução de correções.

Na atividade 6, o processo retorna ao gerente de projetos que, juntamente com o cliente, faz a validação e aprovação dos softwares desenvolvidos. No caso da reprovação do cliente, os requisitos pendentes retornam ao gerente de desenvolvimento para avaliação.

Figura 3 – Processo de desenvolvimento de software na empresa estudada



Fonte: elaborado pelos autores

As variáveis tidas como mais relevantes no processo de desenvolvimento são: (*i*) atendimento às datas de entrega; (*ii*) aderência dos sistemas construídos aos requisitos levantados; e (*iii*) minimização do uso dos recursos humanos. Conforme observado por Pinedo (2008), os itens (*i*) e (*iii*) podem ser diretamente beneficiados através de técnicas de sequenciamento, sendo o primeiro o principal foco deste trabalho.

O setor de desenvolvimento, além das atividades destacadas no processo apresentado na Figura 2, também realiza o suporte avançado a sistemas já em operação. Este suporte consiste no esclarecimento de dúvidas ou problemas de cunho técnico que não puderam ser resolvidos pelo atendimento geral. Como são solicitações não previstas e que requerem prazos curtos de resposta, acabam concorrendo com as atividades de desenvolvimento e tornando o planejamento mais complexo. Neste trabalho, tanto o processo de transformação de requisitos em software quanto à execução de suporte avançado são igualmente denominados como “tarefas”.

O desempenho das heurísticas propostas foi inicialmente avaliado por meio de simulação, como descrito na seção 4.2.

4.2 Resultados da simulação

Os dados simulados foram gerados com base em número de tarefas e tempos de duração usualmente verificados na empresa. O quadro de tarefas levantado apresentou uma fila de execução de 50 tarefas divididas em dois grupos: (*i*) tarefas de Suporte e (*ii*) tarefas de Projeto. A amostra era composta por 20% de tarefas do tipo suporte e 80% do tipo projeto. Os tempos médios de execução de tais atividades, bem como o desvio-padrão, são apresentados na Tabela 1.

Tabela 1 – Parametrização das tarefas utilizadas na simulação

Tipos de tarefa	Número de tarefas para cada cenário	Tempo de execução (em horas)		Prazo de entrega (em dias)	
		Média	Desvio	Média	Desvio
Suporte	10	3	1,7	2	1,4
Projetos	40	10	3,6	30	19

Fonte: Elaborado pelos autores

Além dos dados de tempos de execução e prazos de entrega, foram computadas 0,5 horas para *setup* (observadas quando ocorre alternância entre diferentes projetos ou suporte). As atividades que compreendem este *setup* são: (*i*) envio dos trabalhos realizados para um repositório de códigos; (*ii*) atualização da documentação do projeto que está sendo fechado, e (*iii*) atualização da equipe sobre o andamento e status do projeto/suporte sobre o qual iniciará suas atividades. Por fim, foram considerados 2 times de desenvolvedores.

Na sequência, foram gerados 500 conjuntos com 50 tarefas em cada conjunto, também denominados cenários. Cada cenário teve a duração das tarefas e seus respectivos prazos de entrega gerados aleatoriamente, obedecendo à distribuição normal balizada pelos parâmetros da Tabela 1. O atraso para cada tarefa foi calculado pela diferença entre a data de conclusão e sua respectiva data programada de entrega. É importante ressaltar que sempre foram verificados atrasos nos cenários modelados.

As tarefas de cada cenário foram sequenciadas através das heurísticas H_1 e H_2, gerando valores de atraso total médio e *makespan*. Apesar do *makespan* não consistir na função objetivo primária deste estudo, tal informação oferece apoio de decisão em termos da utilização dos recursos produtivos (PINEDO, 2008).

A heurística H_1 acarretou 33,7 dias de atraso total médio nas 500 replicações de simulação, enquanto H_2 apresentou 25,6 dias (diferença de 31,6% em favor de H_2), conforme apresentado na Tabela 2.

Tabela 2 – Atraso total médio e *makespan* médio nos cenários simulados

Heurística	Atraso Total Médio (em dias)	*Makespan* médio (em dias)
H_1	33,7	72,4
H_2	25,6	72,6

Fonte: Elaborado pelos autores

Além do atraso total médio, contabilizou-se o número de cenários em que cada heurística foi mais eficaz; H_2 apresentou melhores resultados em 88,2% dos casos, H_1 foi melhor em 8,2% dos casos, e em 3,6% os resultados gerados pelas duas heurísticas foram idênticos. Tais resultados são apresentados na Tabela 3.

Em termos de *makespan*, a heurística H_1 apresentou média de 72,4 dias, contra 72,6 dias de H_2. Quando analisado o número de cenários em que cada

heurística foi mais eficaz, H_1 foi melhor em 29,6% dos casos, contra 22,6% de H_2, e resultados iguais em 47,8% das simulações, conforme ilustrado na Tabela 3.

Tabela 3 – Desempenho das heurísticas nos cenários simulados

	Atraso Total Médio		*Makespan*	
	Cenários (número de simulações)	Cenários (%)	Cenários (número de simulações)	Cenários (%)
H_1 melhor	41	8,2%	148	29,6%
H_2 melhor	441	88,2%	113	22,6%
$H_1 = H_2$	18	3,6%	239	47,8%
Total	500	100%	500	100%

Fonte: Elaborado pelos autores

De tal forma, percebe-se que H_2 apresentou resultados satisfatórios para a função objetivo primária desse estudo (minimização do atraso total), porém H_1 conduziu a melhores resultados ao considerar-se a função objetivo secundária (*makespan*). O desempenho computacional de execução das heurísticas foi semelhante, demandando 57 segundos para o processamento de cada cenário.

Os testes foram realizados em um computador do tipo PC Intel Core Duo fabricado em 2008, com sistema operacional Windows 7. Aumentando o número de tarefas de 50 para 100, observou-se que o tempo necessário para a aplicação da heurística H_1 subiu para 131 segundos; similarmente, elevar o número de tarefas para 200 requer 315 segundos de processamento. O comportamento praticamente linear (número de tarefas *versus* tempo de processamento) pode ser comprovado através do teste de hipóstese para verificação da existência de correlação (Montgomery, 2007): F_{cal}=380>>F_{tab}=0,03. Resultados semelhantes foram verificados para a heurística H_2. Tal comportamento corrobora os padrões reportados por Huegler e Vasko (1997).

4.3 Aplicação das heurísticas em um cenário real da empresa

Na sequência, as heurísticas foram aplicadas em tarefas reais coletadas da empresa. Considerou-se um conjunto de tarefas ordenadas manualmente pela gerência da empresa para 2 times de programadores, conforme apresentado na

Figura 4. O atraso total estimado para tal sequência foi de 23 dias, com *makespan* de 73 dias.

Figura 4 – Lista de tarefas ordenada manualmente

Programação de Tarefas para o Time de Desenvolvedores 1			
Tarefa	Tempo de execução (horas)	Projeto	Data desejada entrega
Tarefa 35	10	Projeto 2	11/04/2011
Tarefa 39	3	Suporte	11/04/2011
Tarefa 42	2	Suporte	11/04/2011
Tarefa 49	3	Suporte	12/04/2011
Tarefa 46	2	Suporte	12/04/2011
Tarefa 41	1	Suporte	12/04/2011
Tarefa 40	3	Suporte	13/04/2011
Tarefa 45	2	Suporte	13/04/2011
Tarefa 47	1,5	Suporte	13/04/2011
Tarefa 44	3	Suporte	15/04/2011
Tarefa 22	8	Projeto 2	17/04/2011
Tarefa 33	4	Projeto 2	17/04/2011
Tarefa 37	8	Projeto 2	27/04/2011
Tarefa 36	12	Projeto 2	28/04/2011
Tarefa 25	14	Projeto 2	30/04/2011
Tarefa 26	8	Projeto 2	05/05/2011
Tarefa 30	8	Projeto 2	05/05/2011
Tarefa 24	6	Projeto 2	06/05/2011
Tarefa 19	12	Projeto 2	09/05/2011
Tarefa 38	10	Projeto 2	09/05/2011
Tarefa 23	16	Projeto 2	13/05/2011
Tarefa 31	10	Projeto 2	13/05/2011
Tarefa 27	14	Projeto 2	15/05/2011
Tarefa 29	10	Projeto 2	16/05/2011
Tarefa 32	14	Projeto 2	20/05/2011
Tarefa 20	14	Projeto 2	24/05/2011
Tarefa 34	14	Projeto 2	26/05/2011
Tarefa 21	4	Projeto 2	31/05/2011
Tarefa 28	2	Projeto 2	15/06/2011

Programação de Tarefas para o Time de Desenvolvedores 2			
Tarefa	Tempo de execução (horas)	Projeto	Data desejada entrega
Tarefa 48	4	Suporte	11/04/2011
Tarefa 43	8	Suporte	12/04/2011
Tarefa 5	8	Projeto 1	11/04/2011
Tarefa 13	8	Projeto 1	11/04/2011
Tarefa 2	6	Projeto 1	16/04/2011
Tarefa 9	8	Projeto 1	17/04/2011
Tarefa 18	8	Projeto 1	17/04/2011
Tarefa 10	10	Projeto 1	20/04/2011
Tarefa 1	12	Projeto 1	08/05/2011
Tarefa 7	8	Projeto 1	26/04/2011
Tarefa 4	14	Projeto 1	14/05/2011
Tarefa 12	12	Projeto 1	16/05/2011
Tarefa 16	4	Projeto 1	21/05/2011
Tarefa 17	10	Projeto 1	24/05/2011
Tarefa 3	8	Projeto 1	29/05/2011
Tarefa 11	16	Projeto 1	31/05/2011
Tarefa 6	12	Projeto 1	01/06/2011
Tarefa 15	12	Projeto 1	02/06/2011
Tarefa 8	16	Projeto 1	06/06/2011
Tarefa 14	10	Projeto 1	06/06/2011

Data de início considerada: 11/04/2011

Tempo de *setup* de troca entre projetos: 30 minutos

Carga de trabalho diária: 8 horas/dia

Fonte: Elaborado pelo autor

O mesmo conjunto de tarefas foi então sequenciado pelas heurísticas H_1 e H_2, sendo os resultados apresentados na Tabela 4.

Tabela 4 – Comparação do sequenciamento manual com as heurísticas H_1 e H_2

Forma de sequenciamento	Atraso Total (em dias)	*Makespan* (em dias)
Manual	23	73
Utilizando H_1	23	72
Utilizando H_2	19	74

Fonte: Elaborado pelo autor

A heurística H_1 não alterou o atraso total estimado em relação ao sequenciamento manual, apenas reduziu o *makespan* em um dia, enquanto que a heurística H_2 reduziu o atraso total em 4 dias, o que significa um aprimoramento de 21%.

Por outro lado, foi necessário um dia adicional para conclusão de todas as tarefas, comparando-se com a programação manual. Este comportamento pode ser associado à maior alternância entre projetos, que introduz um maior número de *setups* no processo, mas faz com que as tarefas de menores folgas no cronograma possam ser executadas antes. De tal forma, não existe uma conclusão unânime a respeito da melhor heurística; recomenda-se a aplicação de ambas para assegurar a obtenção de resultados consistentes.

4.4 Implicações gerenciais

Um benefício direto do método proposto é a eliminação do tempo gasto pelo gerente de desenvolvimento na realização do sequenciamento das tarefas. Além disso, a elaboração de uma programação de tarefas deixa de ser responsabilidade exclusiva do gerente, visto que qualquer outro programador pode executar o sequenciamento, dispondo apenas dos tempos de execução e datas de entrega das tarefas.

Também se verificou que os atrasos nas entregas das tarefas passaram a ser identificados com maior facilidade e antecedência, permitindo ações preventivas operacionais e táticas, como o uso de horas-extras ou contratação de novos desenvolvedores.

O reduzido esforço computacional para executar a programação de tarefas permite ainda que simulações de mudanças no ambiente geral da empresa sejam facilmente realizadas. Tais simulações permitem a identificação de cenários otimizados frente a flutuações inesperadas na demanda de projetos.

Como exemplo, foram criados arbitrariamente dois cenários hipotéticos onde a empresa teria assumido a realização imediata de um novo projeto, paralelamente aos projetos consolidados, sendo este projeto composto por 30 novas tarefas (os tempos destas tarefas foram gerados conforme apresentado na seção 4.2). Na

primeira simulação (S1), manteve-se a equipe original de operadores (2 times), o que acarretou significativo aumento no atraso total.

Em uma segunda simulação (S2), ampliou-se a equipe para três times de desenvolvimento, com o intuito de atenuar o problema dos atrasos verificados. Os resultados podem ser observados na Tabela 5.

Tabela 5 - Simulação de novos cenários

	Atraso Total (em dias)	
Heurística	H_1	H_2
Cenário atual	23	19
S1	452	362
S2	99	58

Fonte: Elaborado pelos autores

Verifica-se que a visibilidade obtida por este tipo de simulação ajudou na tomada de decisões estratégicas, como a readequação de times de desenvolvimento ou a mudança na forma de negociação de prazos para novos projetos.

Apesar do presente trabalho ser focado no setor de desenvolvimento de empresas de software, espera-se que a abordagem proposta possa ser aplicada a outros processos produtivos. A indústria calçadista, caracterizada por procedimentos artesanais e com elevada amplitude nos tempos de execução, figura como um potencial setor de aplicação.

5 CONCLUSÕES

O presente trabalho propôs o uso de sequenciamento de tarefas, tipicamente utilizado no setor industrial, para reduzir atrasos nas tarefas de uma empresa de desenvolvimento do software. Duas heurísticas distintas foram apresentadas e aplicadas em cenários que replicavam características semelhantes ao da empresa em estudo. Na sequência, as heurísticas foram aplicadas no cenário real da empresa, e os resultados de atraso total e de *makespan* comparados com o sequenciamento manual realizado pelo gerente responsável.

A heurística H_2 apresentou melhores resultados ao minimizar o atraso total nos dados simulados, ao passo que H_1 conduziu a melhores resultados para a

função objetivo *makespan*. Para o sequenciamento das tarefas reais, a heurística H_2 apresentou melhores resultados considerando-se a minimização do atraso total como função objetivo.

Quando comparadas ao sequenciamento manual, H_2 obteve uma redução de 21% no atraso total, mas aumentou o *makespan* em 1,4%; H_1 manteve o mesmo atraso, porém reduziu o *makespan* em 1,4%.

A similaridade dos resultados gerados pelas heurísticas inibe a recomendação de uma única abordagem. Como o tempo computacional despendido na execução de cada heurística é curto, recomenda-se utilizar as duas heurísticas propostas. De posse dos resultados das duas execuções, se pode fazer uma avaliação mais precisa, inclusive ponderando os resultados oriundos das duas heurísticas em uma decisão multi-criterial.

As seguintes proposições consistem em potenciais desdobramentos deste estudo: (*i*) adição de grau de importância às tarefas a serem sequenciadas; (*ii*) análise do impacto da variação do número de times de desenvolvimento sobre o atraso; (*iii*) adição de tarefas com pré-requisitos; e (*iv*) estudo da curva de aprendizado na velocidade de execução das tarefas pelos times de desenvolvimento.

REFERÊNCIAS

ANZANELLO, M.J.; FOGLIATTO F.S. Scheduling learning dependent jobs in customized assembly lines. **International Journal of Production Research**, v.48, n.22, p.6683-6699, 2010.

APEL, S; KASTNER, C. An overview of feature-oriented software development. **Journal of Object Technology**, v.8, n.5, 49-84, 2009.

ASSOCIAÇÃO BRASILEIRA DAS EMPRESAS DE SOFTWARE (ABES). **Mercado Brasileiro de Software 2011**. Disponível em: <http://www.abes.org.br/templ3 .aspx?id= 306&sub=650>. Acesso em: 07 jul. 2011.

CANEL, C.; ROSEN, D.; ANDERSON, E.A. Just-in-time is not just for manufacturing: a service perspective. **Industrial Management & Data Systems**, v.100, n.2, p.1-60, 2000.

DAVIS, M.M.; AQUILANO, N.J.; CHASE, R.B. **Fundamentos de administração da produção**. 3 ed. São Paulo: Bookman, 2001.

FANDEL, G.; HEGENE, C. S. Simultaneous lot sizing and scheduling for multi-product multi-level production. **International Journal of Production Economics**, v.104, p.308-316, 2006.

FERNANDES, A. A.; TEIXEIRA, D. S. **Fábrica de software:** implantação e gestão de operações. São Paulo: Atlas, 2007.

FRAMINAN, J. M.; RUIZ, R. Architecture of manufacturing scheduling systems: Literature review and an integrated proposal. **European Journal of Operational Research**, v.205, p.237-246, 2010.

HUEGLER, P. A.; VASKO, F. J. A performance comparison of heuristics for the total weighted tardiness problem. **Computers & Industrial Engineering**, v.32, n.4, p.753-767, 1997.

LAM, C. Y.; IP, W. H. Constraint priority scheduling using an agent-based approach. **Industrial Management & Data Systems**, v.111, n.2, p.246-263, 2011.

LIU. M.; CHEN, D.; WU, C.; LI, H. Reduction method based on a new fuzzy rough set in fuzzy information system and its applications to scheduling problems. **International Journal of Computers and Mathematics with applications**, v.51, p.1571-1584, 2006.

MONTGOMERY, D.; RUNGER, G. **Applied statistics and probability for Engineers**, 4th edition, Wiley, 2007.

MOR, B.; MOSHEIOV, G. Batch scheduling on uniform machines to minimize total flow-time. **Computers & Operations Research**, v.39, p.571-575, 2012.

OHNO, T. **O sistema Toyota de produção**: além da produção em larga escala. Porto Alegre: Bookman, 1997.

PINEDO, M. L. **Scheduling:** theory, algorithms, and systems. 3 ed. New York: Springer, 2008.

PROJECT MANAGEMENT INSTITUTE (PMI). A Guide to the Project Management Body of Knowledge **PMBOK Guide**. 3 ed. Pennsylvania: Project Management Institute, 2004.

SOFTWARE ENGINEERING INSTITUTE (SEI). Capability Maturity Model Integration for Development (CMMI-DEV), Carnegie Melon. **Software Engineering Institute**, 2006. Disponível em: <http://www.sei.cmu.edu/cmmi/>. Acesso em: 28 abr. 2011.

SZÖKE, A. Conceptual scheduling model and optimized release scheduling for agile environments. **Information and Software Technology**, v.53, p.574-591, 2011.

THE STANDISH GROUP INTERNATIONAL - CHAOS **Report 2009**. Disponível em: <http://www1.standishgroup.com/newsroom/chaos_2009.php>. Acesso em: 28 abr. 2011.

TUBINO, D.F. **Planejamento e controle da produção**: teoria e prática. São Paulo: Atlas. 2007.

WANG, M. Z.; WANG, J. B. Single machine makespan minimization scheduling with nonlinear shortening processing times. **Computers & Operations Research**, v.39, p.659-663, 2012.

YIN, R.K. **Estudo de caso**: planejamento e métodos. 3 ed. Porto Alegre: Bookman, 2003.

Artigo recebido em 04/08/2011 e aceito para publicação em 02/04/2012.