Minimização dos tempos de atraso na programação de tarefas em uma empresa de desenvolvimento de softwares

"Artigo a ser submetido ao periódico Produção"

Icaro Paulo Ludwig – UFRGS – icaroludwig@gmail.com
Michel José Anzanello – UFRGS – michel.anzanello@gmail.com

Resumo

Empresas de pequeno porte do setor de serviços, como as desenvolvedoras de softwares, usualmente programam suas tarefas de forma manual. Tal programação conduz a resultados satisfatórios 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, conforme o modelo de três estágios abordado por Anzanello e Fogliatto (2010), que (*i*) define um ordenamento inicial para as tarefas; (*ii*) distribui cada tarefa aos times de desenvolvimento; e (*iii*) ordena as tarefas em cada time de desenvolvimento, com vistas ao atendimento do objetivo de minimização do atraso total. Como resultado, se observou que a ferramenta permitiu reduzir os atrasos de entregas, simplificou o processo de programação e deu maior visibilidade do processo de desenvolvimento.

Palavras-chave: Sequenciamento, Atraso total, Setor de software.

Abstract

Small companies in service sectors, such as software developers, usually rely on manual-based programming tasks. That programming works fine for small task lists, but yields management difficulties as task pool size grows, resulting in task delays. This paper aims to use a scheduling tool in order to minimize these delays. For that it proposes two heuristics for task scheduling based on the three stage model of Anzanello and Fogliatto (2010). The proposed approach (i) defines an initial order for tasks, (ii) distributes each task to development teams, and (iii) schedules the tasks in each development team aimed at minimizing total tardiness. The proposed approach reduced total tardiness, simplified the process of scheduling and provided better tracking of the development process.

Keywords: Scheduling, Total tardiness, Software sector

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. Apesar do STP ter raízes no meio industrial, suas ideias têm sido absorvidas por diferentes áreas, como no setor de serviços (CANEL et al., 2000) ou em empresas que desenvolvem projetos, cuja atuação abrange a entrega de produtos conjunta à de serviços, como é o caso de empresas desenvolvedoras de softwares.

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 dentro do esperado (THE STANDISH GROUP, 2009). Tal situação demanda o desenvolvimento de ferramentas robustas para assegurar o cumprimento dos prazos de entrega dos produtos aos clientes.

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 (ETO - *Engineer to Order*). Por conta do elevado nível de customização do setor, parte 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, logo, suas 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. 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, que é peça fundamental na realização de projetos de software personalizados (PARZIANELLO, 2009). Dessa 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 abordadas.

Este trabalho propõe duas heurísticas de programação de tarefas 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 posterior 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 de uma empresa de desenvolvimento de software.

O presente artigo apresenta, na seção 2, os fundamentos da programação de tarefas (também conhecida como sequenciamento de tarefas), 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 obtidos. A seção 5 traz as conclusões do estudo.

2. Referencial Teórico

2.1 O Setor de Software

Segundo a Associação Brasileira das Empresas de Software, o setor de software teve faturamento aproximado de US$ 885 bilhões no mundo durante o ano de 2010. Deste montante, US$ 19 bilhões foram movimentados somente 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 (ABES, 2011), e 76,5% trabalham no desenvolvimento, distribuição e comercialização de softwares; as demais dedicam-se 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, com personalizações 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 (PARZIANELLO, 2009). 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 a codificação, assemelham-se a uma linha de montagem industrial. O uso de ferramentas administrativas, gerenciais e de controle operacional, tipicamente associadas a ambientes industriais, passa a ser adotado nesse tipo de empresa, popularizando o termo fábrica de software. A Figura 1 traz um exemplo de processo de codificação de programas tipicamente utilizado em uma fábrica de software.

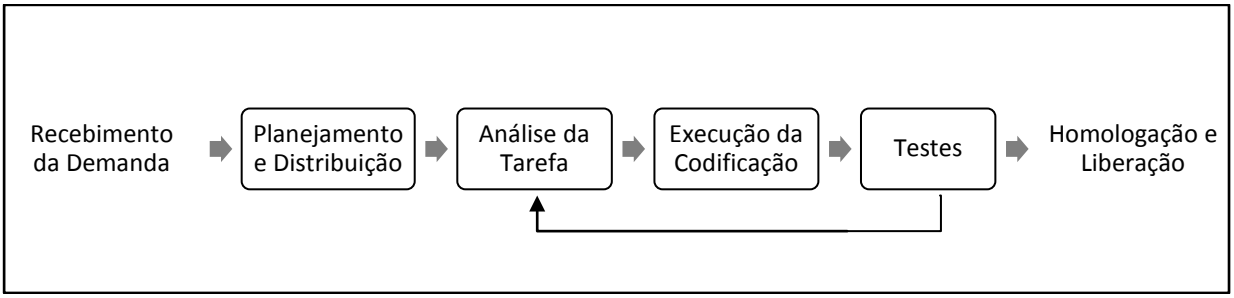

Figura 1. Modelo de Fábrica de Software. Adaptado de Fernandes e Teixeira (2007)

Por conta desta semelhança com linhas de montagem, verifica-se no processo produtivo de software a possibilidade de uso de conceitos e ferramentas originalmente introduzidos pela Engenharia de Produção para a indústria (OHNO, 1997). 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, e que atua de forma tática; e (*iii*) curto Prazo, onde está a Programação da Produção e que tem 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, que 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, se encontram as chamadas funções objetivo do sequenciamento, conforme descritas por 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 todas as tarefa programadas, em relação à data em que a tarefa deveria ter sido entregue; (*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. Através desta função objetivo se pode alimentar indicadores de atendimento ao cliente; 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 estoques de produto acabado.

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

As heurísticas de sequenciamento são normalmente associadas à disposição dos recursos produtivos. Dentre tais disposições, é comum encontrar grupos de recursos formados por máquinas trabalhando em paralelo (PINEDO, 2008). Dentro de tal grupo, é usual encontrar máquinas com capacidades ou tempos de preparação diferentes, onde as ferramentas de sequenciamento devem tratar cada recurso de forma própria. Tais arranjos são chamados de máquinas paralelas não-relacionadas, e figuram entre os mais complexos dentre os 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).

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: $\frac{\text{(data de entrega – } \Sigma \text{ tempo de processamento restante)}}{\text{Número de operações restantes}}$
IFA	Índice de falta	Os lotes serão processados de acordo com o menor valor de: quantidade em estoque / quantidade de demanda

Quadro 1. Regras de Priorização mais utilizadas, segundo Tubino (2007)

Uma técnica sugerida por Pinedo (2008) para o sequenciamento em máquinas paralelas não relacionadas se apoia em dois passos. Primeiramente, determina-se quais tarefas serão alocadas para cada recurso, e depois faz-se o sequenciamento das tarefas dentro da fila de execução de cada um desses recursos. O trabalho de Anzanello e Fogliatto (2010), por sua vez, é focado em trabalhadores, e sugere um passo adicional antes da 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 que a função objetivo seja otimizada.

Especificamente para a função objetivo de minimização do atraso total, a principal dificuldade em implantar ferramentas computacionais de sequenciamento onde se tem um grande número de tarefas é o balanceamento entre a qualidade da solução obtida e o esforço computacional demandado. Os métodos mais simples, como os que aplicam apenas as Regras de Priorização mais utilizadas (TUBINO, 2007) são de rápida execução, mas podem não trazer resultados satisfatórios, sendo denominados "*Quick and Dirty"* por Huegler e Vasko (1997). Heurísticas que apresentam resultados ótimos para essa função objetivo têm elevado esforço computacional, visto que o problema de minimização de atraso total é classificado como *NP-hard* (PINEDO, 2008), ou seja, a hierarquia de complexidade da solução tem um

crescimento dito não polinomial de acordo com o crescimento do número de tarefas a ser sequenciado.

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 reduzido, se comparadas aos métodos tradicionais como o *branch and bound*, e encontram soluções superiores às encontradas pelas Regras de Priorização.

3. Procedimentos Metodológicos

A pesquisa a ser realizada é de natureza aplicada, utiliza uma abordagem quantitativa e tem objetivo exploratório (YIN, 2003). Para tanto é conduzido um estudo de caso da aplicabilidade do sequenciamento em uma empresa de desenvolvimento de software, tipicamente 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 abordado 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) com vistas ao atendimento de uma função objetivo. Esses 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. Nesse 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 será utilizada a regra do tempo de processamento acumulado, cujo objetivo é balancear a carga de trabalho entre os *m* times. Desta forma, cada tarefa *j* será alocada ao time de desenvolvimento que estiver com a menor quantidade de trabalho acumulada até o momento. Essa 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 estágio é utilizado o Algoritmo de Minimização do Atraso Total (AMAT) descrito por Huegler e Vasko (1997) para sequenciar as tarefas em 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 e que objetiva minimizar o somatório do atraso T_j dentro de cada sistema). Essa simplificação visa viabilizar a utilização do AMAT, desenvolvido para sistemas compostos por um único time de desenvolvimento (ou máquina, no contexto de sequenciamento). Tal algoritmo foi selecionado devido à 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 ditas 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 sempre 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.

	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%

Figura 2. Exemplo de aplicação do algoritmo AMAT de Huegler e Vasko (1997)

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. Resolver o problema sequenciando apenas este subconjunto. Caso o subconjunto atual não esteja iniciando na primeira tarefa da lista (*PS*=1), então deve-se 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".

Etapa 3: Aumentar *PS* de [*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 então as últimas *S* tarefas.

Etapa 4: Repetir a execução das 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 ocorra, 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 a serem testadas, com vistas à minimização do atraso total.

Heurística	Estágio 1: Ordenamento inicial das tarefas	Estágio 2: Alocação das tarefas aos times de desenvolvimento	Estágio 3: Sequenciamento das tarefas de 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		

Quadro 2. Resumo das heurísticas utilizadas no sequenciamento

Para avaliação da eficácia das heurísticas propostas, tarefas semelhantes às verificadas na empresa são simuladas, e as heurísticas H_1 e H_2 utilizadas para sequenciar tais tarefas. As heurísticas são então utilizadas no quadro de tarefas reais da empresa e os resultados obtidos comparados aos gerados pelo sequenciamento manual, forma atualmente empregada na empresa. A comparação das heurísticas baseia-se na comparação direta dos atrasos gerados por cada heurística.

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 agrega 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 do 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.

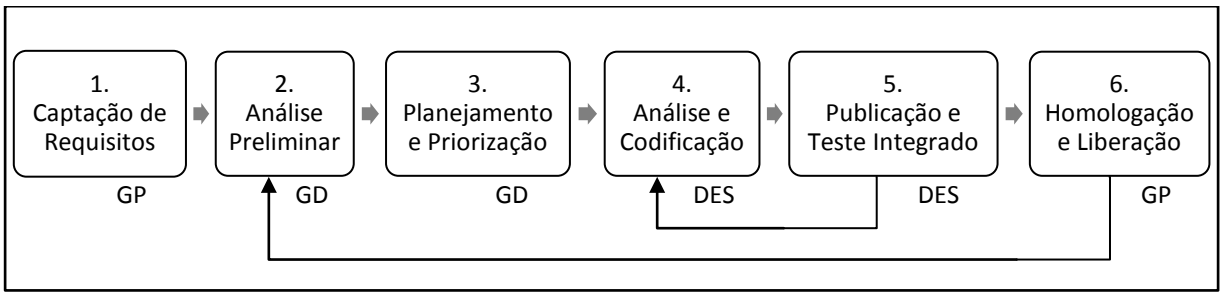

Figura 3. Processo de desenvolvimento de software na empresa estudada

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. Esse 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 a execução de suporte avançado são denominados como tarefas.

O desempenho das heurísticas propostas é avaliado através de simulação e aplicação em dados reais de processo, como segue. Os principais códigos do algoritmo utilizado são apresentados no Apêndice 1.

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 tinha 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. Para as tarefas de suporte os dados históricos apontaram tempos médios de execução de 3 horas (desvio de 1,7 horas) e prazo de entrega médio de 2 dias (desvio de 1,4 dias). As tarefas de projeto apresentaram tempos médios de execução de 10 horas (desvio de 3,6 horas) e prazos de entrega médios de 30 dias (desvio de 19 dias). Um resumo desses dados pode ser observado na Tabela 1.

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

Tabela 1. Parametrização das tarefas utilizadas na simulação

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 esse *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. Foram considerados 2 times de desenvolvedores.

Na sequência, foram gerados 500 conjuntos com 50 tarefas em cada conjunto, também denominados cenários, e cujo objetivo foi reproduzir um possível ambiente que a empresa possa se deparar. Cada cenário teve tanto a duração das tarefas quanto seus respectivos prazos de entrega gerados aleatoriamente, obedecendo à distribuição normal balizada pelos parâmetros da Tabela 1, e o atraso para cada tarefa 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).

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.

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

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

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.

	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%

Tabela 3. Desempenho das heurísticas nos cenários simulados

Em termos de *makespan*, a heurística H_1 apresentou média de 72,4 dias, resultados 0,3% melhores que H_2, que teve 72,6 dias. 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.

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 aos 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 utilizando 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, se observou que o tempo necessário para a aplicação das heurísticas H_1 e H_2 subiu para 131 e 133 segundos, respectivamente. O comportamento praticamente linear do esforço computacional em função do número de tarefas segue os padrões reportados por Huegler e Vasko (1997).

4.3 Aplicação das heurísticas a um cenário real da empresa

Na sequência, as heurísticas foram aplicadas em tarefas reais coletadas da empresa. Partiu-se de uma lista de tarefas ordenada manualmente pela gerência da empresa para 2 times de programadores, conforme apresentado na Figura 4. O atraso total estimado para tal sequência é de 23 dias, com *makespan* de 73 dias.

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

Figura 4. Lista de tarefas ordenada manualmente

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

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

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

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 a mais até a conclusão de todas as tarefas, comparando-se com a programação manual. Esse 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 menor folga no cronograma possam ser executadas antes. De tal forma, não existe uma conclusão unânime a respeito da eficiência das duas heurísticas propostas.

4.4 Implicações gerenciais

Um benefício direto do método proposto é a eliminação do tempo gasto pelo gerente de desenvolvimento na atividade de realizar o sequenciamento, que atualmente consome cerca de 30 minutos semanais. 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.

Além do cunho operacional de organizar a execução das tarefas, também se verificou que os atrasos passam a ser identificados com mais facilidade e maior antecedência, permitindo ações preventivas operacionais, como o uso de horas-extras, e táticas, como a contratação de novos desenvolvedores.

O reduzido esforço computacional para executar a programação de tarefas permite ainda que se façam simulações de mudanças no ambiente geral da empresa. Tais simulações permitem a identificação de cenários otimizados frente a flutuações inesperadas na demanda de projetos. Como exemplo, foram criados arbitrariamente dois cenários fictícios onde a empresa teria assumido a realização imediata de um novo projeto, paralelamente aos projetos consolidados, sendo esse projeto composto por 30 novas tarefas (os tempos destas tarefas foram simuladas 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 gerados. Os resultados podem ser observados na Tabela 5.

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

Tabela 5. Simulação de novos cenários

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

Mesmo o presente trabalho sendo focado no setor de desenvolvimento de empresas de software, espera-se que a abordagem proposta possa ser aplicada a outros processos produtivos organizados conforme o problema de máquinas paralelas não-relacionadas. A indústria calçadista, caracterizada por procedimentos manuais com elevada variabilidade nos tempos de execução, figura como potencial setor de aplicação.

5. Conclusões

O presente trabalho propôs o uso de uma ferramenta de sequenciamento de tarefas, tipicamente utilizada no setor industrial, para reduzir atrasos nas tarefas de uma empresa de desenvolvimento do software. Duas diferentes heurísticas foram apresentadas e aplicadas em cenários que simulavam 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 considerando a função objetivo de minimização de atraso total nos dados simulados, ao passo que H_1 conduziu a melhores resultados para *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 comparados ao sequenciamento manual, H_2 obteve uma redução de 21% no atraso total, mas aumentou em 1,4% o *makespan*, enquanto H_1 manteve o mesmo atraso, porém reduziu 1,4% o *makespan*.

A similaridade dos resultados gerados pelas heurísticas proíbe a seleção de uma única abordagem. Visto que o tempo computacional despendido na execução de cada heurística é de 57 segundos, 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 figuram como 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; (*iv*) adição de velocidades variáveis para cada time de desenvolvimento; (*v*) estudo da curva de aprendizado na velocidade de execução das tarefas pelos times de desenvolvimento; e (*vi*) uso de outras heurísticas de sequenciamento a casos de empresas de serviços.

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.

ASSOCIAÇÃO BRASILEIRA DAS EMPRESAS DE SOFTWARE (ABES). *Mercado Brasileiro de Software 2010*. Disponível em: <http://www.abes.org.br/templ3.aspx?id=306&sub=596>. Acesso em: 07 jul. 2011.

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.

FERNANDES, A.A.; TEIXEIRA, D.S. *Fábrica de Software - Implantação e Gestão de Operações*. São Paulo: Atlas, 2007.

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.

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

PARZIANELLO, L.C. *Uma Visão Ágil do Planejamento e Controle da Produção (PCP) de Software*. Porto Alegre, 2009. (Apostila)

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.

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.

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

Apêndice 1: Algoritmos utilizados na construção das heurísticas propostas

'*** Estágio 1 - Heurística H1

```
Sub a_Ordena_inicial_tamanho()
   Sheets("Ordena_inicial").Select
   Columns("A:Z").Select
   Selection.Delete Shift:=xlToLeft
   Sheets("Tarefas").Select
   Columns("A:G").Select
   Selection.Copy
   Sheets("Ordena_inicial").Select
   Range("A1").Select
   Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks:=False, Transpose:=False
   Selection.PasteSpecial Paste:=xlPasteFormats, Operation:=xlNone, SkipBlanks:=False, Transpose:=False
   Columns("A:G").Select
   ActiveWorkbook.Worksheets("Ordena_inicial").Sort.SortFields.Clear
   ActiveWorkbook.Worksheets("Ordena_inicial").Sort.SortFields.Add Key:=Range("C:C"),
SortOn:=xlSortOnValues, Order:=xlAscending, DataOption:=xlSortNormal
   With ActiveWorkbook.Worksheets("Ordena_inicial").Sort
      .SetRange Range("A:G")
      .Header = xlYes
      .MatchCase = False
      .Orientation = xlTopToBottom
      .SortMethod = xlPinYin
      .Apply
   End With
   Application.CutCopyMode = False
   Range("A1").Select
End Sub
```

'*** Estágio 1 - Heurística H2

```
Sub b_Ordena_inicial_folga()
   Sheets("Ordena_inicial").Select
   Columns("A:Z").Select
   Selection.Delete Shift:=xlToLeft
   Sheets("Tarefas").Select
   Columns("A:G").Select
   Selection.Copy
   Sheets("Ordena_inicial").Select
   Range("A1").Select
   Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks:=False, Transpose:=False
   Selection.PasteSpecial Paste:=xlPasteFormats, Operation:=xlNone, SkipBlanks:=False, Transpose:=False
   Columns("A:G").Select
   ActiveWorkbook.Worksheets("Ordena_inicial").Sort.SortFields.Clear
   ActiveWorkbook.Worksheets("Ordena_inicial").Sort.SortFields.Add Key:=Range("G:G"),
SortOn:=xlSortOnValues, Order:=xlAscending, DataOption:=xlSortNormal
   With ActiveWorkbook.Worksheets("Ordena_inicial").Sort
      .SetRange Range("A:G")
      .Header = xlYes
      .MatchCase = False
      .Orientation = xlTopToBottom
      .SortMethod = xlPinYin
      .Apply
   End With
```

```
   Application.CutCopyMode = False
   Range("A1").Select
End Sub
```

'* Estágio 2 - Heurística H1 e H2**

```
Sub c_Distribui()
   Dim ultima_linha
   Dim contador
   Dim horas_do_time(32)
   Dim numero_times
      numero_times = Range("numero_de_times").FormulaR1C1
   Dim tmo
   Sheets("Distribui").Select
   Columns("A:Z").Select
   Selection.Delete Shift:=xlToLeft
   Sheets("Ordena_inicial").Select
   Columns("A:G").Select
   Selection.Copy
   Sheets("Distribui").Select
   Range("A1").Select
   Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks:=False, Transpose:=False
   Selection.PasteSpecial Paste:=xlPasteFormats, Operation:=xlNone, SkipBlanks:=False, Transpose:=False
   Range("G1").Select
   Selection.Copy
   Range("H1").Select
   Selection.PasteSpecial Paste:=xlPasteFormats, Operation:=xlNone, _
      SkipBlanks:=False, Transpose:=False
   Application.CutCopyMode = False
   ActiveCell.FormulaR1C1 = "Time"
   ' Aloca time
      Range("G1").Select
      Selection.End(xlDown).Select
      ultima_linha = ActiveCell.Row
      For contador = 1 To numero_times
         horas_do_time(contador) = 0
      Next
      For contador = 2 To ultima_linha
         tmo = time_menos_ocupado(horas_do_time(), numero_times)
         Range("H" & contador).FormulaR1C1 = tmo
         horas_do_time(tmo) = horas_do_time(tmo) + Range("C" & contador).FormulaR1C1
      Next
End Sub

'*** Função auxiliar para o Estágio 2

Function time_menos_ocupado(ByRef horas_do_time(), ByVal numero_times) As Integer
   Dim tmo
      tmo = 0
   Dim hmo
      hmo = 99999999
   Dim contador
   For contador = 1 To numero_times
      If horas_do_time(contador) < hmo Then
         tmo = contador
         hmo = horas_do_time(contador)
      End If
```

```
   Next
   time_menos_ocupado = tmo
End Function
```

'* Estágio 3 - Heurística H1 e H2**

```
Sub e_Sequencia()
   Dim retorno
   retorno = executa_sequenciamento()
End Sub
Function executa_sequenciamento()
   Dim retorno
   'Prepara Planilha de Sequenciamento
      linha_sequencia = 2
      Sheets("Sequencia").Select
      Columns("A:Z").Select
      Selection.Delete Shift:=xlToLeft
      Range("E:E").Select
      With Selection.Interior
         .PatternColorIndex = xlAutomatic
         .ThemeColor = xlThemeColorAccent3
         .TintAndShade = 0.8
         .PatternTintAndShade = 0
      End With
      Range("J:J").Select
      With Selection.Interior
         .PatternColorIndex = xlAutomatic
         .ThemeColor = xlThemeColorAccent3
         .TintAndShade = 0.8
         .PatternTintAndShade = 0
      End With
      Range("K:K").Select
      With Selection.Interior
         .PatternColorIndex = xlAutomatic
         .ThemeColor = xlThemeColorAccent3
         .TintAndShade = 0.9
         .PatternTintAndShade = 0
      End With
      Sheets("Ordena_time").Select
      Range("A1:A1").Select
      Selection.Copy
      Sheets("Sequencia").Select
      Range("A1:K1").PasteSpecial Paste:=xlPasteFormats, Operation:=xlNone, SkipBlanks:=False,
Transpose:=False
      Columns("E:E").NumberFormat = "dd/mm/yyyy"
      Columns("G:G").NumberFormat = "dd/mm/yyyy"
      Columns("I:J").NumberFormat = "dd/mm/yyyy"
      Columns("C:K").HorizontalAlignment = xlCenter
      Sheets("Ordena_time").Select
      Range("A1:H1").Select
      Selection.Copy
      Sheets("Sequencia").Select
      Range("A1").Select
      ActiveSheet.Paste
      Range("I1").FormulaR1C1 = "Início Programado"
      Range("J1").FormulaR1C1 = "Final Previsto"
```

```
    Range("K1").FormulaR1C1 = "Atraso (dias)"
    Columns("B:K").EntireColumn.AutoFit
  'sequencia
    retorno = inicia_execucao_seq
  Application.CutCopyMode = False
  Range("A1").Select
  executa_sequenciamento = retorno
End Function


Function inicia_execucao_seq()
  Dim numero_de_times
    numero_de_times = CInt(Range("numero_de_times").FormulaR1C1)
  Dim data_atual As Date
  Dim bloco_tamanho
    bloco_tamanho = CInt(Range("bloco_tamanho").FormulaR1C1)
  Dim bloco_avanco
    bloco_avanco = CInt(Range("bloco_avanco").FormulaR1C1)
  Dim ultima_linha
  Dim i
  Dim tarefa(MAXTAREFAS, 8)
  Dim total_de_tarefas
  Dim tarefas_do_time(MAXTAREFAS, 10)
  Dim tarefas_para_sequenciar(MAXTAREFAS, 10)
  Dim sequencia(MAXTAREFAS)
  Dim sequencia_local(MAXTAREFAS)
  Dim melhor_sequencia(MAXTAREFAS)
  Dim melhor_sequencia_local(MAXTAREFAS)
  Dim melhor_resultado
  Dim melhor_resultado2
  Dim total_do_time
  Dim tempo_setup
    tempo_setup = CDbl(Replace(Range("horas_setup_troca_projeto").FormulaR1C1, ".", ",")) /
CDbl(Replace(Range("horas_por_dia").FormulaR1C1, ".", ","))
  Dim makespan_todos_times
    makespan_todos_times = 0

' Busca indice da última linha
  Sheets("Ordena_time").Select
  Range("G1").Select
  Selection.End(xlDown).Select
  ultima_linha = ActiveCell.Row

'Coloca todas as tarefas numa matriz
  For i = 2 To ultima_linha
    tarefa(i - 1, 1) = Range("A" & i).FormulaR1C1   ' ID
    tarefa(i - 1, 2) = Range("B" & i).FormulaR1C1   ' Descrição
    tarefa(i - 1, 3) = Range("C" & i).FormulaR1C1   ' Tamanho Horas
    tarefa(i - 1, 4) = Range("D" & i).FormulaR1C1   ' Projeto
    tarefa(i - 1, 5) = Range("E" & i).FormulaR1C1   ' Data desejada
    tarefa(i - 1, 6) = Range("F" & i).FormulaR1C1   ' Tamanho em Dias
    tarefa(i - 1, 7) = Range("G" & i).FormulaR1C1   ' Data início mais tarde
    tarefa(i - 1, 8) = Range("H" & i).FormulaR1C1   ' Time de desenv
    total_de_tarefas = i - 1
  Next

'Executa em cada Time de Desenvolvimento
```

```
  Dim time
  For time = 1 To numero_de_times
    melhor_resultado = 999999999
    total_do_time = 0
    For i = 1 To MAXTAREFAS
      sequencia(i) = ""
      melhor_sequencia(i) = ""
      If CInt(tarefa(i, 8)) = time Then
        total_do_time = total_do_time + 1
        tarefas_do_time(total_do_time, 0) = total_do_time
        tarefas_do_time(total_do_time, 1) = tarefa(i, 1)
        tarefas_do_time(total_do_time, 2) = tarefa(i, 2)
        tarefas_do_time(total_do_time, 3) = tarefa(i, 3)
        tarefas_do_time(total_do_time, 4) = tarefa(i, 4)
        tarefas_do_time(total_do_time, 5) = tarefa(i, 5)
        tarefas_do_time(total_do_time, 6) = tarefa(i, 6)
        tarefas_do_time(total_do_time, 7) = tarefa(i, 7)
        tarefas_do_time(total_do_time, 8) = tarefa(i, 8)
        tarefas_do_time(total_do_time, 9) = ""
        tarefas_do_time(total_do_time, 10) = ""
      End If
    Next

    For ciclos_completo = 1 To MAXCICLOS
      data_atual = CDate(Range("data_inicio_trabalhos").FormulaR1C1)
      primeira_tarefa = 1 - bloco_avanco

      'Processo iterativo Top-Down (do primeiro subconjunto para o último)
        Do
          'calcula indices inicial e final para sequenciamento
            primeira_tarefa = primeira_tarefa + bloco_avanco
            ultima_tarefa = primeira_tarefa + bloco_tamanho - 1
            If ultima_tarefa > total_do_time Then
              ultima_tarefa = total_do_time
            End If
          'monta matriz do subconjunto para sequenciar
            melhor_resultado_local = 999999999
            melhor_resultado_local2 = 999999999
            For j = 1 To MAXTAREFAS
              sequencia_local(j) = ""
              melhor_sequencia_local(j) = ""
            Next
            j = 0
            For i = primeira_tarefa To ultima_tarefa
              j = j + 1
              tarefas_para_sequenciar(j, 0) = j
              For k = 1 To 10
                tarefas_para_sequenciar(j, k) = tarefas_do_time(i, k)
              Next
            Next
          'sequencia a submatriz
            aaa = ordena_recursiva(tarefas_para_sequenciar(), sequencia_local(), 1, j,
melhor_sequencia_local(), melhor_resultado_local, melhor_resultado_local2, data_atual)
            aaa = ordena_melhor_sequencia(tarefas_para_sequenciar(), melhor_sequencia_local())
          'atualiza dados da matriz do subconjunto sequenciado de volta no conjunto principal
            j = 0
            For i = primeira_tarefa To ultima_tarefa
```

```
                j = j + 1
                For k = 1 To 10
                  tarefas_do_time(i, k) = tarefas_para_sequenciar(j, k)
                  melhor_sequencia(i) = melhor_sequencia_local(j)
                Next
              Next
            'recalcula todas as datas
              data_atual = CDate(Range("data_inicio_trabalhos").FormulaR1C1)
              aaa = atribuir_datas(tarefas_do_time(), melhor_sequencia(), data_atual)
            'ajusta dados de retorno (data_atual, resultados, sequencias)
              If (primeira_tarefa + bloco_avanco - 1) < ultima_tarefa Then
                data_atual = tarefas_do_time(primeira_tarefa + bloco_avanco - 1, 10)    'pegou a data de fim da
tarefa antes do inicio do próximo bloco
              End If
          Loop While (ultima_tarefa < total_do_time)
        'Fim do Processo iterativo Top-Down

        'Processo iterativo Bottom-Up (do último subconjunto para o primeiro)
          ultima_tarefa = total_do_time
          Do
            'calcula indices inicial e final para sequenciamento
              primeira_tarefa = primeira_tarefa - bloco_avanco
              If primeira_tarefa < 1 Then
                primeira_tarefa = 1
              End If
              ultima_tarefa = primeira_tarefa + bloco_tamanho - 1
              If ultima_tarefa > total_do_time Then
                ultima_tarefa = total_do_time
              End If
            'atualiza informacao da data_atual (data de inicio considerada para este sequenciamento)
              If primeira_tarefa = 1 Then
                data_atual = CDate(Range("data_inicio_trabalhos").FormulaR1C1)
              Else
                data_atual = tarefas_do_time(primeira_tarefa - 1, 10)
                If tarefas_do_time(primeira_tarefa, 4) <> tarefas_do_time(primeira_tarefa - 1, 4) Then
                  data_atual = data_atual + tempo_setup
                End If
              End If

            'monta matriz do subconjunto para sequenciar
              melhor_resultado_local = 999999999
              melhor_resultado_local2 = 999999999
              For j = 1 To MAXTAREFAS
                sequencia_local(j) = ""
                melhor_sequencia_local(j) = ""
              Next
              j = 0
              For i = primeira_tarefa To ultima_tarefa
                j = j + 1
                tarefas_para_sequenciar(j, 0) = j
                For k = 1 To 10
                  tarefas_para_sequenciar(j, k) = tarefas_do_time(i, k)
                Next
              Next
            'sequencia o subconjunto
              aaa = ordena_recursiva(tarefas_para_sequenciar(), sequencia_local(), 1, j,
melhor_sequencia_local(), melhor_resultado_local, melhor_resultado_local2, data_atual)
```

```
            aaa = ordena_melhor_sequencia(tarefas_para_sequenciar(), melhor_sequencia_local())
          'atualiza dados do subconjunto sequenciado de volta na matriz principal
            j = 0
            For i = primeira_tarefa To ultima_tarefa
              j = j + 1
              For k = 1 To 10
                tarefas_do_time(i, k) = tarefas_para_sequenciar(j, k)
                melhor_sequencia(i) = melhor_sequencia_local(j)
              Next
            Next
          'recalcula todas as datas, considerando offsets
            data_atual = CDate(Range("data_inicio_trabalhos").FormulaR1C1)
            aaa = atribuir_datas(tarefas_do_time(), melhor_sequencia(), data_atual)
        Loop While (primeira_tarefa > 1)

      'Fim do Processo iterativo Bottom-Up
    Next

    'escreve dados do time na planilha do sequenciamento
      escreve_sequencia tarefas_do_time(), melhor_sequencia()

    'adiciona makespan de cada time ao total
      makespan_todos_times = makespan_todos_times + aaa

  Next  'fecha ciclo de sequenciamento de cada time

  inicia_execucao_seq = makespan_todos_times
End Function


'*** Função recursiva de sequenciamento

Function ordena_recursiva(ByRef tarefas_do_time(), ByRef sequencia(), ByVal ini, ByVal total_do_time, ByRef
melhor_sequencia(), ByRef melhor_resultado, ByRef melhor_resultado2, ByVal data_atual)
  Dim i
  Dim retorno
  Dim retorno2
  Dim temp
  Dim aaa
  For i = ini To total_do_time '(posicao ma matriz)
    For tar_atual = 1 To total_do_time '(tarefa em questão)
      'grava tarefa na posicao
        sequencia(i) = tarefas_do_time(tar_atual, 1)
      'se sequencia completa, então calcula
        If i = total_do_time Then
          'Verifica se a sequencia é valida
          If valida_sequencia(sequencia(), total_do_time) Then
            aaa = atribuir_datas(tarefas_do_time(), sequencia(), data_atual)
            retorno = calcula_tard_mksp(sequencia(), total_do_time, tarefas_do_time(), retorno2)
            If retorno < melhor_resultado Then
              For j = 1 To MAXTAREFAS
                melhor_sequencia(j) = sequencia(j)
              Next
              melhor_resultado = retorno
              melhor_resultado2 = retorno2
            ElseIf (retorno = melhor_resultado) And (retorno2 < melhor_resultado2) Then
              For j = 1 To MAXTAREFAS
```

```
                    melhor_sequencia(j) = sequencia(j)
                  Next
                  'melhor_resultado = retorno
                  melhor_resultado2 = retorno2
                End If
              End If
        'senão, chama novamente ordena_recursiva
          Else
            temp = ordena_recursiva(tarefas_do_time(), sequencia(), (ini + 1), total_do_time,
melhor_sequencia(), melhor_resultado, melhor_resultado2, data_atual)
          End If
      Next
   Next
   ordena_recursiva = True
End Function

'*** Função de cálculo do atraso

Function calcula_tard_mksp(ByRef sequencia(), ByVal total_do_time, ByRef tarefas_do_time(), ByRef retorno2)
   retorno2 = 0
   Dim i
   Dim j
   Dim retorno As Double
     retorno = 0
   Dim atraso_tarefa
   Dim prj_anterior
     prj_anterior = ""
   Dim prj_atual
     prj_atual = ""
   Dim tempo_setup_horas
     tempo_setup_horas = CDbl(Replace(Range("horas_setup_troca_projeto").FormulaR1C1, ".", ",")) '/
CDbl(Replace(Range("horas_por_dia").FormulaR1C1, ".", ","))

   'algoritmo de calculo do TOTAL TARDINESS com segunda opção por MAKESPAN
     For j = 1 To UBound(sequencia)
       If sequencia(j) = "" Then
         Exit For
       Else
         'TARDINESS (Função objetivo principal)
           i = busca_nas_tarefas_do_time(tarefas_do_time, sequencia(j), 0)
           atraso_tarefa = CDate(tarefas_do_time(i, 10)) - CDate(tarefas_do_time(i, 5))
           If atraso_tarefa > 0# Then
             retorno = retorno + atraso_tarefa
           End If
         'MAKESPAN (Função objetivo como critério de desempate)
           'Verifica necessidade de SETUP
             prj_atual = tarefas_do_time(i, 4)
             If prj_atual <> prj_anterior Then
               'Testa se não é SETUP
                 If prj_anterior <> "" Then
                   retorno2 = retorno2 + tempo_setup_horas
                 End If
               prj_anterior = prj_atual
             End If
           'Adiciona horas das tarefas
             retorno2 = retorno2 + tarefas_do_time(i, 3)
       End If
```

```
      Next
   calcula_tard_mksp = retorno
End Function

'*** Função de cálculo das datas iniciais e finais de cada tarefa

Function atribuir_datas(ByRef matriz_tarefas(), ByRef melhor_sequencia(), ByVal data_atual)
   Dim horas_por_dia
      horas_por_dia = Range("horas_por_dia").FormulaR1C1
   Dim numero_dias
   Dim dia_da_semana
   Dim considera_sabado
      considera_sabado = Range("considera_sabado").FormulaR1C1
   Dim tempo_setup
      tempo_setup = CDbl(Replace(Range("horas_setup_troca_projeto").FormulaR1C1, ".", ",")) /
CDbl(Replace(Range("horas_por_dia").FormulaR1C1, ".", ","))
   Dim i
   Dim j
   Dim prj_anterior
      prj_anterior = ""

   For j = 1 To UBound(melhor_sequencia)
      If melhor_sequencia(j) = "" Then
         Exit For
      Else
         i = busca_nas_tarefas_do_time(matriz_tarefas, melhor_sequencia(j), 0)
         'Verifica necessidade de SETUP
            If prj_anterior <> matriz_tarefas(i, 4) Then
               'Testa se não é SETUP
                  If prj_anterior <> "" Then
                     data_atual = data_atual + tempo_setup
                  End If
               prj_anterior = CStr(matriz_tarefas(i, 4))
            End If
         'Verifica se não caiu no final de semana
            dia_da_semana = Weekday(data_atual)
            'Descarta domingo
               If dia_da_semana = 1 Then
                  data_atual = data_atual + 1#
               End If
            'Descarta sabado se neessário
               If dia_da_semana = 7 And CInt(considera_sabado) <> 1 Then
                  data_atual = data_atual + 2#
               End If
         'Seta data inicial da tarefa
            matriz_tarefas(i, 9) = data_atual

         'Adiciona tempo de execução da tarefa
            numero_dias = CDbl(Replace(matriz_tarefas(i, 3), ".", ",")) / horas_por_dia
            Do While (numero_dias > 0#)
               If numero_dias >= 1# Then
                  data_atual = data_atual + 1
               Else
                  data_atual = data_atual + numero_dias
               End If
               dia_da_semana = Weekday(data_atual)
               'Descarta domingo
```

```
                If dia_da_semana = 1 Then
                    data_atual = data_atual + 1#
                End If
            'Descarta sabado se necessário
                If (dia_da_semana = 7) And (CInt(considera_sabado) <> 1) Then
                    data_atual = data_atual + 2#
                End If
            numero_dias = numero_dias - 1
        Loop
      'Grava data de fim da tarefa
        matriz_tarefas(i, 10) = data_atual
    End If
  Next
  atribuir_datas = DateDiff("d", CDate(Range("data_inicio_trabalhos").FormulaR1C1), data_atual)
End Function
```