Desmistificando o Desenvolvimento de Software *Follow-the-Sun:* Caracterização e Lições Aprendidas

Josiane Kroll, Jorge L. N. Audy, Rafael Prikladnicki

Faculdade de Informática – PUCRS
90619-900 – Porto Alegre – RS – Brasil
josiane.kroll@acad.pucrs.br,{audy,rafaelp}@pucrs.br

Abstract. *Follow-the-Sun (FTS) is a software development strategy applied to global software development by 24 continuous hours. However, the FTS is few practiced by companies and sometimes misunderstood. Thus, with purpose to provide a conceptual basis to FTS implementation, this paper presents the characterization and distinction of the FTS based on the comparison of others definitions found in the literature. In addition, we also present learned lessons with case studies already conducted in the area.*

Resumo. *O Follow-the-Sun (FTS) é uma estratégia de desenvolvimento de software global que é aplicada para o desenvolvimento de software por 24 horas contínuas. No entanto, o FTS é pouco praticado pelas empresas e muitas vezes até mal compreendido. Dessa forma, com o propósito de fornecer uma base conceitual para a aplicação do FTS, neste artigo é apresentada a caracterização e distinção do FTS com base na comparação de outras definições encontradas na literatura. Além disso, apresentamos também lições aprendidas com estudos de caso conduzidos na área.*

1. Introdução

O *Follow-the-Sun* (FTS) é uma estratégia de *Global Software Development* (GSD)[1] que se caracteriza pelo desenvolvimento de software por 24 horas contínuas. O seu principal objetivo é reduzir o tempo de produção do produto para colocação no mercado *(time-to-market)* [Carmel Espinosa e Dubinsky 2010]. No entanto, o FTS ainda é pouco praticado pelas empresas e muitas vezes até mal compreendido, devido as dificuldades que há em implementá-lo. Na literatura, também não existe uma uniformidade de processos e conceitos entre os autores que abordam o assunto.

Nesse trabalho, com o propósito de fornecer uma base conceitual para a aplicação da estratégia FTS pelas empresas, são apresentadas as características que definem o FTS e é realizada uma comparação entre os demais conceitos encontrados na literatura e que são referenciados com o mesmo sentido. Além disso, a partir de alguns estudos de caso publicados, foram identificadas lições aprendidas com a aplicação da estratégia FTS. A contribuição deste artigo está em fornecer um melhor entendimento e

[1] O *Global Software Development* (GSD) é também referenciado na literatura como *Global Software Engineering* (GSE), *Distributed Software Engineering* (DSE) ou Desenvolvimento Distribuído de Software (DDS) [Prikladnicki e Audy 2010] [Jabangwe e Nurdiani 2010]. O termo será mantido em inglês, visto que a tradução em alguns casos pode comprometer o uso do termo.

desmistificação do FTS, que apesar de ser uma estratégia complexa e de difícil execução, pode trazer vantagens significativas no contexto de GSD.

Este artigo está organizado da seguinte forma: na Seção 2, é apresentado o referencial teórico da área de estudo, incluindo a comparação entre os termos encontrados na literatura com a caracterização do FTS. Na Seção 3, são apresentadas as lições aprendidas. Por fim, na Seção 4 as considerações finais do trabalho apresentado.

2. Referencial Teórico

2.1. A estratégia *Follow-the-Sun* (FTS)

O FTS é uma estratégia de desenvolvimento de software que se caracteriza pelo desenvolvimento 24 horas contínuas com equipes geograficamente distribuídas [Visser e Solingen 2009]. O FTS considera as diferenças de tempo e a comunicação contínua para definir turnos e horários de trabalho, com reuniões regulares e relatórios de progresso, a fim de identificar e aplicar estratégias adequadas do início ao final do projeto [Chinbat 2010].

O FTS é utilizado principalmente para estruturar o desenvolvimento de software de acordo com o tempo, permitindo a redução do tempo de desenvolvimento do produto [Herbsleb 2007]. Consequentemente, o conhecimento projetado pelo FTS também objetiva reduzir a duração do projeto, aumentar a produção e melhorar a alocação de tarefas nas 24 horas de desenvolvimento [Sooraj e Mohapatra 2008].

Na estratégia FTS as equipes são distribuídas globalmente de maneira que quando uma equipe encerra suas horas normais trabalho a outra equipe localizada em outro local e com o fuso horário diferente assume as tarefas iniciando sua jornada de trabalho. A produção diária realizada por uma equipe segue então para o próximo local que pode estar em um diferente fuso horário para ser continuado por outra equipe de trabalho [Visser e Solingen 2009].

A continuidade do trabalho envolve ciclos de troca de tarefas entre as equipes, que são separados por *handoffs* diários [Carmel Dubinsky e Espinosa 2009] [Solingen e Valkema 2009]. O *handoff* é o termo utilizado na literatura para designar o processo de transição de tarefas entre equipes e pode ser definido como um *check-in* da unidade de trabalho para o próximo local de produção [Carmel Dubinsky e Espinosa 2009].

As dificuldades encontradas para a aplicação da estratégia FTS envolvem principalmente os desafios de comunicação, coordenação de processos, gerência de equipes e diferenças culturais e geográficas [Carmel Dubinsky e Espinosa 2009]. No entanto, no desenvolvimento FTS também são observadas vantagens que podem ser obtidas em relação aos demais tipos de desenvolvimento de software aplicados em ambiente de GSD. Essas vantagens foram definidas por Carmel, Espinosa e Dubinsky (2010), sendo que para as mesmas serem obtidas é necessário que sejam satisfeitos 4 critérios:

1. O principal objetivo do FTS é a redução do tempo de desenvolvimento do software para colocação do produto no mercado *(time-to-market):* Esse critério distingue o FTS das demais práticas e configurações do GSD tradicional. O FTS não oferece outras vantagens sobre outras configurações, além da velocidade de desenvolvimento.

2. Os locais de desenvolvimento devem estar distantes entre si e em diferentes fusos horários. Esse critério diferencia o FTS dos demais tipos de GSD pelo aumento da velocidade do desenvolvimento.

3. Em qualquer ponto no tempo, apenas um local possui o produto. Esse critério diferencia o FTS a partir das configurações convencionais de GSD, em que vários locais podem possuir diferentes partes do produto.

4. *Handoffs* são realizados diariamente no final de cada turno. Este critério diferencia o FTS das configurações convencionais de GSD que minimiza as dependências de *handoffs* entre locais de desenvolvimento.

2.2. Conceitos similares para FTS identificados na literatura

Na literatura foram encontrados quatro conceitos similares ao FTS, quais sejam: *24-hour development model* [Jalote e Jain 2004], *24-Hour Knowledge Factory Paradigm (24HrKF)* [Denny Crk e Sheshu 2008], *Desenvolvimento round-the-clock* [Carmel Dubinsky e Espinosa 2009] [Visser e Solingen 2009] e *Software Shift Work* [Gorton, Hawryszkiewycz e Fung 1996].

O *24-hour development model* é definido como uma abordagem para o GSD, onde uma equipe distribuída trabalha em diferentes fusos horários e locais de produção para estabelecer um *workflow* de 24 horas em um único projeto. Seu objetivo é reduzir o tempo total de desenvolvimento do projeto, através de *handoffs* diários de trabalho [Jalote e Jain 2004]. Essa abordagem considera o *24-Hour Knowledge Factory Paradigm* (24HrKF), onde múltiplas equipes estão distribuídas em diferentes fusos horários de trabalho em um mesmo projeto [Denny Crk e Sheshu 2008]. No *24-hour development model*, o potencial de melhoria do tempo de desenvolvimento do projeto, que consiste na redução do mesmo, é fortemente dependente do grau de interdependência entre as tarefas do projeto e da natureza das várias restrições que podem ocorrer em relação ao ambiente de desenvolvimento.

O *24-Hour Knowledge Factory Paradigm* (24HrKF) sugere equipes geograficamente dispersas, onde os membros da equipe são capazes de trabalhar em processos específicos em um desenvolvimento *round-the-clock*. Uma equipe de trabalho pode estar trabalhando em um fuso horário diferente da outra equipe de trabalho. No final do dia de trabalho, a tarefa é transferida para outra equipe que está localizada em um diferente fuso horário, que por sua vez, dará continuidade a mesma tarefa. Cada equipe trabalha durante o horário de expediente normal que pertence ao seu fuso horário. O 24HrKF pode ser definido como um conjunto do conhecimento de uma equipe com a produção do conhecimento baseado no ativo, com equipes criando ativos incrementais que são os *handoffs* entre equipes de desenvolvimento [Gupta et al. 2009].

O desenvolvimento *round-the-clock* é um método utilizado para a redução do tempo de duração de um projeto, no qual o conhecimento que envolve o produto é de propriedade do local de produção. O *handoff* produzido no final de cada dia de trabalho é passado para o próximo local de produção que está em um diferente fuso horário [Carmel Espinosa e Dubinsky 2010]. Embora a ideia do desenvolvimento *round-the-clock* seja promissora, ainda há poucas evidências da sua eficiência, especialmente quando implementado para alta granularidade de tarefas [Visser e Solingen 2009].

O conceito *Software Shift Work* surgiu devido às significativas diferenças de tempo entre as equipes de desenvolvimento de software. O *Software Shift Work* consiste no desenvolvimento de um produto por 24 horas contínuas. O desenvolvimento de software por 24 horas contínuas explora as diferenças de horário entre equipes, dando o efeito de trabalho por turnos, mas permitindo que as equipes trabalhem em seus horários convencionais de trabalho conforme suas localizações. O objetivo do *Software Shift Work* é reduzir significativamente o tempo de desenvolvimento do produto no mercado [Gorton Hawryszkiewycz e Fung 1996].

2.3. Uma análise comparativa dos conceitos apresentados

Na literatura são encontrados muitos trabalhos que utilizam diferentes definições, não havendo uma uniformidade em caracterizar o FTS. Dessa forma, na Tabela 1 é apresentada uma comparação entre as características identificadas nos quatro conceitos similares ao FTS e no conceito dado por Carmel Dubinsky e Espinosa (2009) ao termo *Follow-the-Sun (FTS).*

Tabela 1. Quadro comparativo dos conceitos de FTS na literatura

Características	***24-hour development model***	**24HrKF**	**Desenvolvimento *round-the-clock***	***Software Shift Work***	***FTS***
Abordagem para o GSD	X	X	X	X	X
Opera com equipes distribuídas trabalhando em diferentes fusos horários e locais de desenvolvimento	X	X	X	X	X
Permite o desenvolvimento 24 horas contínuas de um único projeto	X	X	X	X	X
Objetiva reduzir o tempo total de desenvolvimento de um projeto	X		X		
Objetiva reduzir o tempo total de desenvolvimento do produto		X		X	
Objetiva reduzir o tempo total de colocação do produto no mercado *(time-to-market)*					X
Possui *handoffs* diários de trabalho	X	X	X		X
Dependente do *handoff* para dar continuidade ao trabalho					X
Reduz custos de desenvolvimento	X	X	X		
Possui interdependência entre as tarefas	X	X	X	X	
Cada equipe trabalha com processos específicos	X	X	X	X	
Produz ativos incrementais	X	X			
As equipes trabalham dando continuidade nas mesmas tarefas					X
Em qualquer ponto no tempo, apenas um local possui o produto.					X
Conhecimento que envolve o produto é de propriedade do local de desenvolvimento	X	X	X		

Com as informações apresentadas na Tabela 1 é possível observar que o FTS se distingue dos demais conceitos encontrados na literatura devido a quatro principais características:

1. Tem por objetivo reduzir o tempo total de colocação do produto no mercado (time-to-market): O FTS tem como principal objetivo reduzir o tempo total de colocação do produto no mercado e tem como vantagem principal a velocidade de desenvolvimento [Carmel Espinosa e Dubinsky 2010]. Isso significa que o principal benefício do FTS é a redução do tempo de desenvolvimento. A redução do tempo de desenvolvimento e consequentemente a colocação do produto no mercado é muito importante considerando que na indústria os produtos tornam-se obsoletos rapidamente. Os demais termos encontrados na literatura objetivam principalmente a redução do tempo de desenvolvimento do produto, do projeto e a redução de custos [Gorton Hawryszkiewycz e Fung 1996] [Carmel Espinosa e Dubinsky 2010] [Jalote e Jain 2004] [Gupta Mattarelli Seshasai e Broschak 2009]. Isso faz sentido se observado que o processo de desenvolvimento de software que é definido por esses conceitos são diferentes do FTS.

2. *Dependente do handoff para dar continuidade ao trabalho:* No FTS são realizados *handoffs* diários de trabalho no final de cada dia [Holmstrom et al. 2006]. A equipe que está trabalhando em uma tarefa inicia o seu dia de trabalho com o *handoff* da equipe anterior. Esta por sua vez, irá produzir um novo *handoff* no final do seu dia de trabalho para a próxima equipe que está em um diferente local de desenvolvimento [Carmel Espinosa e Dubinsky 2009]. Dessa forma, uma equipe depende do *handoff* diário da outra equipe para dar continuidade à tarefa. Na definição do 2*4-hour development model, 24-Hour Knowledge Factory Paradigm* (24HrKF) e Desenvolvimento *round-the-clock* é possível observar que não há uma dependência de *handoffs* da equipe anterior para dar continuidade a tarefa. Isso porque nessas definições as tarefas são individuais. No *Software Shift Work* não é especificada a presença de *handoffs* diários de trabalho, embora ocorra um processo incremental no desenvolvimento do software.

3. As equipes trabalham dando continuidade nas mesmas tarefas: Essa característica define um aspecto importante do processo de desenvolvimento de software proposto pela estratégia FTS. As equipes distribuídas globalmente estarão trabalhando sempre na mesma tarefa até que a mesma seja finalizada [Visser e Solingen 2009]. Não há uma divisão ou atribuição de atividades para serem cumpridas por cada equipe, e sim a presença do *handoff* que documenta o que a equipe realizou naquele dia de trabalho e o que a outra equipe deve ter conhecimento para dar continuidade ao processo de desenvolvimento de software.

4.*Em qualquer ponto no tempo, apenas um local possui o produto:* O produto de software no FTS sempre estará de posse de uma equipe de trabalho em um local de desenvolvimento. A transferência do produto de software para outra equipe será sempre acompanhando de um *handoff* com as especificações necessárias para dar continuidade às tarefas de desenvolvimento do software. Nos métodos 2*4-hour development model, 24-Hour Knowledge Factory Paradigm* (24HrKF) e Desenvolvimento *round-the-clock* o conhecimento agregado ao produto de software é de propriedade do local de desenvolvimento e não é transferido para outras equipes que estão em diferentes locais e que participam do mesmo projeto [Visser e Solingen 2009].

Essas quatro características ajudam a definir o FTS e diferenciá-lo dos demais termos encontrados na literatura e que são referenciados com o mesmo sentido. A análise dos resultados obtidos com a comparação dos termos também contribui para

identificar pontos importantes do FTS que devem ser levados em consideração para a aplicação da estratégia FTS.

3. Lições Aprendidas

Os estudos conduzidos nas empresas IBM [Treinen e Miller-Frost 2006] [Carmel Dubinsky e Espinosa 2009], Infosys [Carmel 2006], HP e Intel [Conchúir et al. 2006] mostram muitas características da aplicação do FTS. Com base nesses estudos, lições aprendidas foram identificadas para a aplicação do FTS.

Lição 1: *É muito importante que a distribuição do trabalho cubra fusos horários diferentes e aloque tempo suficiente para a realização de cada tarefa:* De acordo com Treinen e Miller-Frost (2006) o GSD deve ser planejado de maneira a minimizar seus desafios. As diferenças de fusos horários presentes no GSD e exploradas positivamente pelo FTS devem se adequar a distribuição de tarefas nos locais de desenvolvimento. Nesse sentido, deve-se ter cuidado ao distribuir as tarefas entre as equipes que estão em diferentes fusos horários e locais. Não existe uma metodologia específica para a distribuição de tarefas entre as equipes, mas trabalhos como de Visser e Solingen (2009) ajudam a simular as diferenças de fusos horários entre as equipes e indicar locais mais apropriados para o desenvolvimento FTS.

Lição 2: *No início do projeto é necessário identificar as características dos locais de desenvolvimento:* Muitos projetos de GSD fracassam pela falta de planejamento [Jabangwe e Nurdiani 2010]. No estudo de caso realizado na IBM [Treinen e Miller-Frost 2006] um dos motivos que levou ao fracasso de um projeto de GSD que utilizava a estratégia FTS foi a falta de infraestrutura e acessibilidade nos locais de desenvolvimento. As diferenças culturais das equipes alocadas para o projeto, a infraestrutura do local e os demais fatores que levam a minimização dos desafios do GSD devem ser considerados no planejamento de um projeto de GSD que utiliza a estratégia FTS. Treinen e Miller-Frost (2006) recomendam principalmente definir processos de gerenciamento de código fonte e de infraestrutura.

Lição 3: *Os handoffs de trabalho devem ter o nível adequado de detalhamento das atividades para que sejam compreendidos por outras equipes que estão em diferentes locais*: Os *handoffs* são uma das principais dificuldades de coordenação encontradas no FTS [Sooraj e Mohapatra 2008] [Carmel Dubinsky e Espinosa 2009]. Dessa forma, é necessário que as informações contidas em um *handoff* sejam suficientes, de modo que a próxima equipe ao iniciar sua jornada de trabalho tenha todas as informações necessárias para dar continuidade ao processo de desenvolvimento de software [Treinen e Miller-Frost 2006]. Carmel, Dubinsky e Espinosa (2009) citam que a utilização de um repositório pequeno e comum, com partes de código, testes e *check-in* diários pode melhorar o processo de *handoff* e facilitar o trabalho das equipes.

Lição 4: *A aplicação do FTS em apenas algumas fases do projeto de software pode retornar bons resultados:*No estudo realizado por Carmel (2006) na empresa Infosys, foi constatado que a mesma empregava o FTS somente em algumas fases do projeto e obtinha sucesso. As fases escolhidas para a aplicação do FTS dependiam das circunstâncias do projeto. A Infosys utilizava o FTS para reduzir o tempo do projeto e desconsiderava a utilização do FTS com um todo no projeto, devido aos desafios de comunicação, coordenação e cultura. No estudo conduzido na empresa Intel, concluiu-se que o FTS deve ser mais praticado em diferentes fases do desenvolvimento de

software, como por exemplo, a fase de testes, para trazer melhores resultados [Conchúir et al. 2006].

Lição 5: *O FTS reduz o tempo de resposta entre os locais de desenvolvimento se possuir no máximo equipes distribuídas em apenas dois fusos-horário diferentes:* Para a empresa HP o tempo de resposta entre os locais de desenvolvimento é uma fase crítica do ciclo de vida de desenvolvimento de software [Conchúir et al. 2006]. Dessa forma para minimizar esse problema, a HP distribui as equipes em somente dois fusos-horário diferentes, com o propósito de facilitar o gerenciamento. A estratégia adotada pela HP faz sentido, se observados os desafios de comunicação, coordenação e cultura que podem ser minimizados.

4. Considerações Finais

A partir dos resultados apresentados foi possível observar que os conceitos encontrados na literatura são diferentes, embora referenciados em alguns trabalhos como similares. A caracterização do FTS é dada principalmente por quatro características: O FTS tem o objetivo de reduzir o tempo total de colocação do produto no mercado, é dependente de *handoffs* para dar continuidade ao trabalho, onde as equipes trabalham dando continuidade nas mesmas tarefas e em qualquer ponto no tempo, apenas um local possuirá o produto de software. Também cinco lições aprendidas foram identificadas.

Em trabalhos futuros, pretende-se investigar as práticas do FTS empregadas em cada fase do ciclo de desenvolvimento de software no contexto do GSD. Além disso, embora o FTS ainda seja recente como tema de pesquisa, percebe-se uma boa oportunidade para explorá-la no contexto do Brasil. Estudos recentes apontam o país como um potencial candidato a realizar projetos *offshore*, devido a vantagens tais como o fuso horário [Brasscom 2009]. Neste contexto, as empresas podem optar por utilizar os serviços providos pelo Brasil e Índia juntos, utilizando a estratégia FTS de forma a reduzir significativamente a duração de seus projetos, aproveitando um período do dia onde as equipes estariam descansando.

Referências

Brasscom (2009). “Brazil IT-BPO Book 2008-2009”, Brasscom, São Paulo, 2009.

Carmel, E. (2006) “Building your Information Systems from the Other Side of the World: How Infosys manages time differences”, In: Management Information Systems Quarterly – Executive.

Carmel, E.; Dubinsky, Y. e Espinosa, J. A. (2009) “Follow-The-Sun Software Development: New Perspectives, Conceptual Foundation, and Exploratory Field Study”, In: Proceedings of the 42nd Hawaii International Conference on System Sciences.

Carmel, E.; Espinosa, A. J. e Dubinsky, Y. (2010) "Follow the Sun: Workflow in Global Software Development”, In: Journal of Management Information Systems Vol. 27 No. 1, 17 – 38.

Chinbat, S. (2010) “Lessons Learned in Virtual Teams from Global Software Development”, In: Report/Department of Applied Information Technology, University of Gothenburg, Gothenburg, Sweden.

Conchúir, E. Ó; Holmström, H.; Agerfalk, P. J. e Fitzgerald, B. (2006) "Global Software Development: Never Mind the Problems – Are There Really Any Benefits?", In: Proc. 29th Information Systems Research Seminar in Scandinavia.

Denny, N.; Crk, I. e Seshu, R. (2008) “Agile Software Processes for the 24-Hour Knowledge Factory Environment”, In: Journal of Information Technology Research, Vol. 1, Issue 1, 57-71.

Gorton, I.; Hawryszkiewycz, I. e Fung, L. (1996) “Enabling software shift work with groupware: a case study”, In: Proceedings of the Twenty-Ninth Hawaii International Conference on, vol.3,72-81.

Gupta, A.; Mattarelli, E.; Seshasai, S. e Broschak, J. (2009) “Use of collaborative technologies and knowledge sharing in co-located and distributed teams: Towards the 24-h knowledge factory”, In: The Journal of Strategic Information Systems, Volume 18, 147-161.

Herbsleb, J. D. (2007) “Global Software Engineering: The Future of Socio-technical Coordination”, In: IEEE International Conference on Global Software Engineering (ICGSE).

Holmstrom, H.; Conchuir, E. O.; Agerfalk, P. J. e Fitzgerald, B. (2006) “Global Software Development Challenges: A Case Study on Temporal, Geographical and Socio- Cultural Distance”, In: Proceedings of the IEEE international conference on Global Software Engineering (ICGSE '06), Washington, DC, USA, 3-11.

Jabangwe, R. e Nurdiani, I. (2010) “Global Software Development Challenges and Mitigation Strategies: A Systematic Review and Survey Results,” Master´s program in Software Engineering, Blekinge Institute of Technology, OM/School of Computing.

Jalote, P. e Jain, G. (2004) "Assigning tasks in a 24-hour software development model", In: Software Engineering Conference,11th Asia-Pacific, 309- 315.

Prikladnicki, R. e Audy, J. L. N. (2010) "Process Models in the Practice of Distributed Software Development: A Systematic Review of the Literature". Information and Software Technology, v. 1, 779-791

Prikladnicki, R.; Damian, D. e Audy, J. L. N. (2008) “Patterns of Evolution in the Practice of Distributed Software Development: Quantitative Results from a Systematic Review”. Evaluation and Assessment in Software Engineering, Bari, Italy, pp. 1–10.

Solingen, V. R. e Valkema, M. (2010) "The Impact of Number of Sites in a Follow the Sun Setting on the Actual and Perceived Working Speed and Accuracy: A Controlled Experiment" In: 5th IEEE International Conference on Global Software Engineering (ICGSE), 165- 174.

Sooraj, P. e Mohapatra, P. K. J. (2008) "Modeling the 24-h software development process", In: Strategic Outsourcing: An International Journal, 122 – 141.

Treinen, J. J. e Miller-Frost, S. L. (2006) “Following the Sun: Case Studies in Global Software Development”, In: IBM Systems Journal, Volume 45, Number 4.

Visser, C. e Solingen R. V. (2009) “Selecting Locations for Follow-the-Sun Software Development: Towards a Routing Model”. In: Fourth IEEE International Conference on Global Software Engineering.