Follow-the-Sun: Um Processo para Minimizar as Dificuldades de Projetos que Adotam esta Estratégia

Estevão R. Hess, Jorge L. N. Audy

Programa de Pós-Graduação em Ciência da Computação – Pontifícia Universidade Católica do Rio Grande do Sul (PUC/RS)

estevao.hess@acad.pucrs.br, audy@pucrs.br

Abstract. *Despite the challenges of strategy FTS, this way of developing software arouses the interest of the industry, because through this strategy, it is possible to reduce time-to-market, thus increasing their productivity. Therefore, it becomes increasingly important to find ways to alleviate these difficulties. Thus, this work presents a proposal for a process to handle the daily work hand-off during the development phase in a software project.*

Resumo. *Apesar dos desafios da estratégia FTS, esta forma de desenvolver software desperta o interesse da indústria, pois através desta estratégia, é possível diminuir o time-to-market, aumentando assim, a sua produtividade. Portanto, torna-se cada vez mais importante encontrar maneiras de atenuar estas dificuldades. Neste sentido, este trabalho apresenta uma proposta de um processo para a transferência diária de trabalho durante a fase de desenvolvimento de um projeto de software.*

1. Introdução

Atualmente, o processo de globalização está se destacando e gerando grande desafio para a área de engenharia de software (ES). Cada vez mais os projetos estão sendo desenvolvidos em ambientes geograficamente distribuídos, caracterizando o desenvolvimento distribuído de software (DDS).

Durante a utilização do DDS surgem diversos desafios de gerenciamento, sendo a diferença de fuso horário apontada por alguns autores como um fator de extrema relevância [Holmstrom et al. 2006, Treinen et al. 2006]. A estratégia *follow-the-sun* (FTS) surge para atenuar as adversidades causadas pela diferença de fuso horário, usando esta diferença como uma vantagem [Holmstrom et al. 2006, Treinen et al. 2006, Carmel et al. 2009, Lings et al. 2007, Setamanit et al. 2007, Solingen e Valkema 2010, Knob 2007]. Entretanto, estudos mostram que são poucos os casos de sucesso na indústria utilizando esta estratégia [Solingen e Valkema 2010, Carmel et al. 2009]. Os principais problemas apontados pela literatura estão relacionados as dificuldades de sincronização, coordenação e comunicação, principalmente durante a transferência de trabalho de um centro de desenvolvimento para outro [Carmel et al. 2009, Setamanit et al. 2007, Solingen e Valkema 2010].

A utilização da estratégia FTS em todas as fases do ciclo de vida do desenvolvimento do software pode-se apresentar muito complexa. Entretanto, a utilização dentro de uma fase particular é mais adequada para a estratégia FTS, pois as suas características específicas permitem uma estrutura mais controlada para as transferências de trabalho (*hand-offs*) [Carmel et al. 2009, Carmel et al. 2010]. Neste sentido, a literatura apresenta uma lacuna nesta área. Portanto, este trabalho propõe um processo de transferência de trabalho para atenuar as dificuldades relacionadas ao uso desta estratégia. Este processo está focado exclusivamente na fase de desenvolvimento do ciclo de vida de

software. O processo proposto utiliza algumas práticas já adotadas em metodologias ágeis tais como *Test-driven development* (TDD) aplicadas a projetos que utilizam a estratégia FTS. Este estudo torna-se relevante, pois a criação de um processo para a transferência de trabalho pode facilitar o uso desta estratégia nos projetos de software. Além disto, para a teoria da área, esta pesquisa torna-se importante, pois a literatura não apresenta a definição de um processo para a transferência de trabalho em projetos que utilizam esta estratégia.

Este artigo está assim estruturado: a seção 2 apresenta o desenvolvimento distribuído de software; a seção 3 mostra a estratégia *follow-the-sun*; a seção 4 descreve os trabalhos relacionados; a seção 5 apresenta o processo proposto; a seção 6 apresenta os trabalhos futuros; e, finalmente, a seção 7 apresenta as conclusões deste trabalho.

2. Desenvolvimento Distribuído de Software

O DDS surgiu nos anos 90, onde as empresas começaram a desenvolver software com equipes de desenvolvimento distribuídas [Lane et al. 2008]. O DDS é caracterizado sempre que um ou mais recursos envolvidos no projeto estiver fisicamente distante dos demais [Audy e Prikladnicki 2007]. Quando a distância física entre os elementos dos times do projeto abrange mais de um país, caracteriza-se o Desenvolvimento Global de Software (GSD, do inglês *Global Software Development*) [Lane et al. 2008]. Recentemente, as companhias estão utilizando cada vez mais esta forma de desenvolver software, devido as vantagens que o DDS pode prover. A Tabela 1 apresenta alguns destes fatores motivadores citados na literatura, juntamente com as referências.

Tabela 1. Fatores motivadores do uso do DDS

Fator motivador	Referências
Redução de custos	[Lane et al. 2008, Damian et al. 2006, Prikladnicki et al. 2008, Audy e Prikladnicki 2007, Martignoni et al. 2009, Knob 2007]
Ganho de proximidade com o cliente	[Lane et al. 2008, Knob 2007]
Redução do tempo de projeto / *time-to-market*	[Damian et al. 2006, Prikladnicki et al. 2008, Carmel et al. 2009]
Recursos especializados e globais	[Lane et al. 2008, Martignoni et al. 2009]

Apesar de todas as vantagens que o DDS disponibiliza para as organizações, o processo de desenvolvimento de software continua sendo uma atividade complexa. Utilizando o DDS, adiciona-se ao processo alguns desafios, como a distância física, diferenças de fusos horários e diferenças culturais [Prikladnicki 2009, Damian et al. 2006], os quais tornam este tipo de projeto extremamente complexo de ser gerenciado. Segundo alguns autores, os desafios encontrados no DDS podem ser divididos em algumas categorias [Audy e Prikladnicki 2007, Prikladnicki 2009, Knob 2007], as quais afetam determinadas áreas do processo. A Tabela 2 apresenta alguns destes desafios, categorizados conforme a literatura desta área, juntamente com as suas referências.

Tabela 2. Desafios Impostos pelo DDS

Categoria	Desafio	Referências
Organização	Legislação	[Karolak 1998]
	Gerenciamento de portfólio de projeto	[Prikladnicki 2009]
Projetos	Arquitetura de software	[Prikladnicki 2009, Bosch et al. 2010]
	Processos de desenvolvimento	[Audy e Prikladnicki 2007, Prikladnicki 2009]
	Telecomunicações	[Audy e Prikladnicki 2007]
	Gerenciamento de projetos	[Prikladnicki 2009]
Pessoas	Confiança	[Oshri et al. 2007, Prikladnicki 2009]
	Diferenças culturais	[Bin 2007]
	Diferenças de fusos horários	[Audy e Prikladnicki 2007, Carmel et al. 2009]

3. Estratégia *Follow-the-Sun*

Dentre os diversos desafios impostos pelo DDS, o desenvolvimento FTS surge para atenuar os desafios relacionados a diferença de fuso horário. Através da utilização da estratégia FTS, é possível reverter o desafio em relação a diferença de fuso horário em uma vantagem para o projeto [Carmel et al. 2009, Holmstrom et al. 2006, Lings et al. 2007, Setamanit et al. 2007, Solingen e Valkema 2010, Knob 2007, Treinen et al. 2006]. Porém, devido ao fato do desenvolvimento FTS ser uma área nova de estudo na engenharia de software, pouco sobre esta temática foi publicado [Treinen et al. 2006]. Casos de sucesso na indústria utilizando o FTS ainda são escassos [Solingen e Valkema 2010, Carmel et al. 2009].

Após análise da literatura na temática da estratégia de desenvolvimento FTS, identificou-se a falta de um consenso para uma definição para este conceito. Entretanto, alguns trabalhos apresentam definições distintas para este tema [Visser 2009, Gorton et al. 1996, Gupta et al. 2009, Treinen et al. 2006, Carmel et al. 2009]. Após a análise destas definições, propomos uma definição para o desenvolvimento FTS, a qual sintetiza as idéias básicas deste conceito, e pode ser descrita da seguinte forma:

O *follow-the-sun* é uma estratégia de desenvolvimento global de software onde o principal objetivo é a diminuição do *time-to-market,* acelerando a construção do produto final desde a concepção até a sua distribuição. Este ambiente opera com equipes distribuídas em fusos horários e países distintos, onde cada equipe detém o trabalho por determinado período, até que o mesmo seja transmitido para a próxima equipe que inicia a sua jornada. A transferência pode ser para qualquer tipo de tarefa relacionada com o desenvolvimento do projeto de software. Esta transferência deve acontecer diariamente e de forma padronizada.

3.1. Pesquisas em *Follow-the-Sun*

A literatura apresenta alguns estudos relacionados à utilização da estratégia FTS. Estes estudos relatam experimentos comparando a utilização do uso da estratégia FTS com o desenvolvimento tradicional de software [Setamanit et al. 2007, Carmel et al. 2009].

Após análise das pesquisas na área, contata-se que grande parte das dificuldades está na transferência de trabalho de um centro de desenvolvimento para outro, também chamado de *hand-off*. Para atenuar estes desafios, surge a necessidade da criação de um processo para esta transferência de trabalho. A seguir apresenta-se alguns trabalhos relacionados à este tema, para então propor um processo para a transferência de trabalho durante a fase de desenvolvimento do ciclo de vida do desenvolvimento de software.

4. Trabalhos Relacionados

A literatura atual ainda carece de trabalhos relacionados com a estratégia FTS. Publicações que apresentam formas para a utilização da estratégia FTS ainda são escassas. Entretanto temos trabalhos que versam com uma temática semelhante a este estudo.

O trabalho proposto por Lindemann *et al.* [Fadel et al. 2000], mostra formas para acelerar o tempo de desenvolvimento de um projeto. Para alcançar tal objetivo, os autores distribuíram equipes através de diferentes fusos horários e fizeram o uso da estratégia FTS. Criou-se um processo de transferência de trabalho de um *site* para outro. Este processo consistiu em alocar trinta minutos (da equipe terminando o turno de trabalho e a que está começando) para preparar as informações a serem utilizadas durante o *hand-off*. Neste momento de trabalho simultâneo, todos os artefatos criados são entregues, assim como qualquer informação relevante para dar continuidade ao trabalho. Esta comunicação é realizada de forma síncrona, utilizando recursos de telefonia. Logo após, a equipe que iniciou o dia de trabalho, realiza um *brain storm*, onde o estado atual do trabalho é

discutido. Baseado no trabalho que ainda deve ser realizado, as tarefas são alocadas para todos os recursos dentro da equipe. Chegando ao final do dia, este processo é repetido. Este ciclo encerra-se no momento em que os requisitos estão todos alcançados.

O trabalho publicado por Taweel *et al.* [Taweel et al. 2002] apresenta os resultados de um experimento para avaliar a viabilidade do uso de um processo seqüencial colaborativo de engenharia de software para ambientes distribuídos em diferentes fusos horários. Neste experimento foi desenvolvida uma simples calculadora. O projeto foi dividido em 3 fases: *set-up*, onde foi apresentado todos os requisitos para todas as equipes, juntamente com a distribuição do trabalho e o prazo para conclusão; *execução*, onde ocorreu a implementação usando os times distribuídos; *finalização*, onde os dados do experimento foram coletados. O processo colaborativo avaliado estava baseado no envio de e-mails entre as equipes com o status atual do projeto, contendo todas as informações relativas ao trabalho realizado. O trabalho mostra que, apesar de tratar apenas de tarefas simples, os resultados demonstram a viabilidade deste processo.

O trabalho apresentado por Denny *et al.* [Denny et al. 2008] apresenta o conceito de *Composite Personae* (CP). Este conceito mostra como equipes distribuídas podem trabalhar como um único time virtual. Para que isto aconteça, é importante ter uma equipe coesa, espalhada por diferentes fusos horários. Desta forma, o trabalho pode ser passado de um *site* para outro, e o mesmo é continuado. Todo o trabalho está baseado em *hand-off*, onde uma equipe termina o seu dia de trabalho e outra inicia. Porém, alguns problemas podem ocorrer nesta transferência. Para tanto, este trabalho mostra uma forma simples de transição de trabalho. Esta transição está baseada em reuniões de *stand-up*, oriundas do S*crum*. Ao se aproximar do final de um dia de trabalho, os desenvolvedores devem adicionar seus resultados no repositório de código, e preencher um formulário automatizado, chamado de ferramenta de *hand-off*. Neste formulário, deve-se responder as três perguntas básicas de uma reunião S*tand-up*:

i. Quais tarefas foram realizadas desde a última reunião?
ii. O que está planejando realizar até a próxima reunião?
iii. Existe algum problema impedindo você de realizar seu objetivo?

Após preencher estas informações, o trabalho é considerado entregue para a equipe seguinte. O próximo *site* inicia o dia de trabalho coletando as informações disponibilizadas pelo *site* anterior e definindo o que deve ser realizado, utilizando como principal referência, as respostas para as perguntas *i, ii* e *iii* respondidas pelo site anterior. O trabalho [Denny et al. 2008] destaca a importância de ter equipes equivalentes em todos os *sites* distribuídos. Esta equivalência não está relacionada ao número de recursos em cada *site*, mas em capacidade de entrega e de solução de problemas.

Denny et al. [Denny et al. 2009] apresentam um processo de transferência de conhecimento, criado especialmente para o uso do conceito de fábrica de conhecimento em ambientes distribuídos. Este processo foi criado com base no *Personal Software Process* (PSP) [Humphrey et al. 1995]. Este processo é desenvolvido para facilitar a transferência de conhecimento de um time para outro ao final de cada dia (*hand-off*). O trabalho apresenta ainda algumas formas para facilitar o entendimento do trabalho entre as equipes distribuídas. Uma destas formas é através da técnica de *Test-Driven Development* (TDD). Segundo os autores, TDD indica a utilização de testes unitários automatizados para redução de defeitos e controle de qualidade. Nesta técnica, os casos de teste são escritos com o objetivo de validar se todos os requisitos estão implementados da forma correta. Os casos de teste tornam-se um registro documentado da compreensão do requisito e da solução encontrada para atender o mesmo [Denny et al. 2009].

A principal diferença entre os trabalhos relacionados e o trabalho que está sendo proposto está no momento e na forma como a transferência de trabalho deve ser realizada. O processo que está sendo proposto está focado exclusivamente na fase de

desenvolvimento. Outra diferença importante é com o processo proposto pelo trabalho [Taweel et al. 2002] onde as tarefas que cada equipe distribuída irá desenvolver são definidas a priori, ao invés de tratar o time inteiro como um único time virtual.

5. Processo Proposto

Este processo visa atenuar os desafios de coordenação, sincronização e comunicação durante a transferência de trabalho durante a fase de desenvolvimento do ciclo de vida. Neste sentido, os principais objetivos deste processo são:

a. Ao iniciar um dia de trabalho, uma equipe deve, de forma simples, ter a percepção do trabalho que deve ser desenvolvido e o trabalho já realizado.
b. Evitar a necessidade de comunicação síncrona entre equipes distribuídas.
c. Garantir que a transferência de trabalho de um centro de desenvolvimento para o outro ocorra sem problemas.

Processos desta natureza ainda são insuficientes na literatura. Entretanto, alguns trabalhos mostram que este tipo de processo deve ser "leve", ou seja, não pode acarretar um grande *overhead* em um dia típico de trabalho [Denny et al. 2009, Taweel et al. 2002].

Conforme apresentado na seção 4, o processo que está sendo proposto utiliza como base o *Composite Persona* (CP) apresentado por [Denny et al. 2008] e o processo chamado *24hr Design and Development*, apresentado por [Fadel et al. 2000]. Porém, este processo atua apenas durante a fase de desenvolvimento do ciclo de vida de desenvolvimento de software. Apesar de atuar apenas durante esta fase, algumas pré-condições das fases anteriores (definição de requisitos e projeto) auxiliam para o funcionamento do processo:

1. Definição dos requisitos: esta fase tem como artefato de saída, a documentação contendo os requisitos do sistema a ser desenvolvido. Para o bom funcionamento do processo, é importante que os requisitos sejam definidos da forma mais específica possível, preferencialmente utilizando o conceito de *User Stories* [Gupta et al. 2009, Haugen 2006], as quais dividem os requisitos em pequenas funcionalidades para diminuir a complexidade das tarefas [Fadel et al. 2000, Denny et al. 2008]. Este passo torna-se importante, pois é crucial que toda a equipe de desenvolvimento tenha o total entendimento do trabalho a ser realizado [Taweel et al. 2002].

2. Projeto: os artefatos que esta fase irá produzir estão relacionados diretamente com a maneira como as funcionalidades serão implementadas. Diagramas necessários para o entendimento do sistema, definição de classes e dos métodos necessários para que a funcionalidade seja desenvolvida são alguns exemplos do resultado desta fase. Ainda, baseado nos critérios de aceitação oriundos da fase anterior, os testes unitários devem ser criados, para fazer o uso da técnica de *Test-Driven development* (TDD), a qual é a base deste processo. O TDD ainda está relacionado ao fato de manter um registro documentado da compreensão do requisito e da solução encontrada para atender o mesmo [Denny et al. 2009, Gupta et al. 2009].

Conforme ilustrado na Figura 1, a fase de desenvolvimento inicia neste ponto. Os próximos cinco estados, enumerados na Figura 1, representam um único dia de trabalho de uma determinada equipe. Este processo é incremental e estes cinco estados serão repetidos a cada dia de trabalho, para cada time de desenvolvimento distribuído [Fadel et al. 2000, Gupta et al. 2009]. Cada estado pode ser sucintamente descrito da seguinte forma:

1. Este estado marca o início de um dia de trabalho de uma equipe. Carrega-se a versão mais recente do código-fonte. Gera-se um relatório com os testes que já estão e os que ainda não estão aceito. Ou seja, se o teste unitário está "passando", significa que aquele critério de aceitação já está coberto, e não é necessário trabalhar no mesmo. Ainda, a equipe deve analisar o formulário com as informações fornecidas pelo site anterior. Este formulário está baseado em reuniões de *stand-up*, oriundas do *Scrum* [Denny et al. 2008, Gupta et al. 2009].

2. Após análise das informações que o site anterior disponibilizou, a equipe que inicia o seu dia de trabalho deve reunir-se e fazer a distribuição das tarefas (planejamento diário). Esta distribuição deve levar em consideração as informações fornecidas pelo site anterior, assim como o resultado dos testes unitários fornecidos pelo site anterior [Denny et al. 2008, Fadel et al. 2000, Denny et al. 2009].

3. Esta etapa apenas marca a implementação dos requisitos, seguindo a ordem de prioridades definidas no passo 2. Nesta etapa, a equipe foca no desenvolvimento das funcionalidades.

4. Chegando ao final do dia, cada membro da equipe deve reservar um tempo para fazer o *check-in* do código e preencher o formulário de *hand-off*, com todas as informações necessárias para o próximo site. Este formulário está baseado no formato de reuniões *stand-up*, oriundas do *Scrum*, cujo objetivo é responder as três perguntas: *(i)* Quais tarefas foram realizadas desde a última reunião?; *(ii)* O que está planejando realizar até a próxima reunião?; *(iii)* Existe algum problema impedindo você de realizar seu objetivo? [Taweel et al. 2002, Denny et al. 2008, Gupta et al. 2009]. Este formulário é utilizado para formalizar a transferência de trabalho [Denny et al. 2008, Fadel et al. 2000].

5. *Hand-off* concluído: está etapa apenas marca o final de um dia de trabalho. Novos critérios de aceitação estão cobertos, o código fonte mais recente está no repositório e a documentação necessária para o início do trabalho do próximo time está disponível. A transferência de trabalho está concluída [Carmel et al. 2009, Carmel et al. 2010, Taweel et al. 2002, Fadel et al. 2000].

Estes cinco passos são repetidos até que todos os critérios de aceitação estão cobertos, ou seja, todos os testes criados durante a fase de *projeto* estão cobertos. Após cobrir todos os critérios de aceitação encerra-se a fase de desenvolvimento e temos a funcionalidade implementada [Fadel et al. 2000, Gupta et al. 2009].

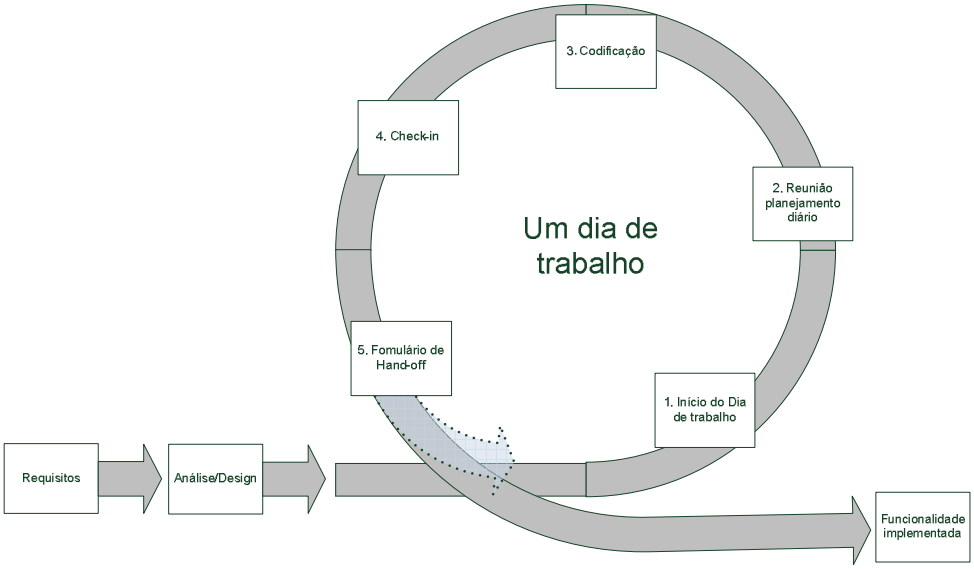

Figura 1. Processo Proposto

6. Trabalhos Futuros

Os próximos trabalhos relacionados ao processo proposto visam a experimentação do processo proposto. Após obter resultados destes experimentos, deve-se identificar os pontos de falhas e o processo deve ser refinado. Ainda, torna-se relevante realizar experimentos com mais centros de desenvolvimento distribuídos, para avaliar se o processo proposto é escalável de forma satisfatória. Trabalhos com o objetivo de expandir este processo para

outras fases dos projetos de software são relevantes. Desta forma, outras fases do SDLC poderiam ser contempladas com a criação de um processo para aumentar a adoção do FTS. Focando em fases especificas de diferentes formas, ao final, pode-se criar um processo, composto por diversos sub-processos, os quais contemplariam todas as fases do SDLC. Desta forma, todo o projeto de software poderia ser realizado utilizando a estratégia FTS.

7. Conclusão

Os objetivos do processo proposto neste trabalho estavam focados na criação de uma proposta de um processo que suavize os desafios impostos pelo uso da estratégia FTS. Com o processo proposto, a contribuição deste trabalho situa-se basicamente em duas dimensões: para a teoria e para o mercado. Para a teoria, a contribuição está na criação de uma proposta de um processo de transferência de trabalho para projetos que utilizam a estratégia FTS, pois a literatura da área não apresenta processo semelhante, exclusivamente para a fase de desenvolvimento. Para o mercado, notamos que, atualmente, na busca de vantagens competitivas, como a diminuição de custo e o ganho de produtividade, as indústrias estão realizando operações de *offshore*. Neste sentido, este trabalho pode contribuir com o aumento do ganho de produtividade, já que irá facilitar o uso da estratégia FTS durante a fase de desenvolvimento, diminuindo, assim, o tempo gasto durante esta fase do ciclo de vida.

Referências

Audy, J.; Prikladnicki,R.,"Desenvolvimento Distribuído de Software – Desenvolvimento de Software com Equipes Distribuídas."Brasil:Campus,2007. 232p.

Holmstrom, Helena; Conchuir, Eoin O; Agerfalk, Par J; Fitzgerald, Brian; , "Global Software Development Challenges: A Case Study on Temporal, Geographical and Socio-Cultural Distance," Global Software Engineering, 2006. ICGSE '06. International Conference on , vol., no., pp.3-11, Oct. 2006.

Treinen, James J., Miller-Frost, Susan L. "Following the Sun : Case Studies in Global Software Development". In : IBM Systems Journal, Volume 45, Number 4, October 2006.

Carmel, E.;Espinosa, A.; Dubinsky, Y., "Follow The Sun Software Development : New Perspectives , Conceptual Foundation , and Exploratory Field Study," 42nd Hawaii International Conference on System Sciences, Proceedings , 2009.

Lings B., Lundell B., Ågerfalk P. J., Fitzgerald B., "A reference model for successful Distributed Development of Software Systems," Global Software Engineering, 2007. ICGSE 2007. IEEE International Conference on , vol., no., pp. 130-139. 2007.

Setamanit, S.-o.; Wakeland, W.; Raffo, D.; , "Improving Global Software Development Project Performance Using Simulation,"Management of Engineering and Technology,Portland International Center for,vol. no., pp.2458-2466, 5-9 Aug. 2007.

van Solingen, Rini; Valkema, Menno; , "The Impact of Number of Sites in a Follow the Sun Setting on the Actual and Perceived Working Speed and Accuracy: A Controlled Experiment," Global Software Engineering (ICGSE), 2010 5th IEEE International Conference on , vol., no., pp.165-174, 23-26 Aug. 2010.

Carmel, E.; Espinosa, A.; Dubinsky, Y., "Follow the Sun" Workflow in Global Software Development Journal of Management Information Systems Vol. 27 No.1, pp. 17 – 38, 2010.

Haugen, N.C.; , "An empirical study of using planning poker for user story estimation," Agile Conference, 2006 , vol., no., pp.9 pp.-34, 23-28 July 2006.

Denny, Nathan, Crk, Igor, Nadella, Ravi Sheshu and Gupta, Amar, "Agile Software Processes for the 24-Hour Knowledge Factory Environment" (February 27, 2009)

Taweel, Adel, Brereton, Pearl: "Developing Software Across Time Zones: An Exploratory Empirical Study". Informatica (Slovenia) 26(3): 2002.

Lingampally, Raghu; Gupta, Atul; Jalote, Pankaj; , "A Multipurpose Code Coverage Tool for Java," System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on , vol., no., pp.261b, Jan. 2007.

Humphrey, W. S. (1995) "Introducing the personal software process", Annals Of Software Engineering volume 1, 1995.

Denny, Nathan, Mani, Shivram, Sheshu Nadella, Ravi, Swaminathan, Manish, Samdal, Jamie: "Hybrid Offshoring: Composite Personae and Evolving Collaboration Technologies". IRMJ 21(1): 89-104, 2008.

Fadel, G., Lindemann, U., Anderl, R.; "Multi-National Around the Clock Collaborative Senior Design Project", Invited paper, Honorable mention at the ASME Curriculum Innovations Award 2000.

Lane, M.; Agerfalk, P., "On the Suitability of Particular Software Development Roles to Global Software Development", Global Software Engineering, 2008. ICGSE 2008. IEEE International Conference on , vol., no., pp.3-12, 17-20 Aug. 2008.

Damian, D.; Moitra, D., "Guest Editors' Introduction: Global Software Development: How Far Have We Come?", Software,IEEE,vol.23,no.5,Sept.-Oct., p.17-19, 2006.

Prikladnicki, R., "Padrões de Evolução na Prática de Desenvolvimento de Software em Ambientes de Internal Offshoring: Um Modelo de Capacidade." Tese de Doutorado, PUCRS, 2009.

Prikladnicki, R.; Damian, D.; Audy, J., "Patterns of Evolution in the Practice of Distributed Software Development in Wholly Owned Subsidiaries: A Preliminary Capability Model," Global Software Engineering, 2008. ICGSE 2008. IEEE International Conference on , vol., no., pp.99-108, 17-20 Aug. 2008.

Martignoni, R., "Global Sourcing of Software Development - A Review of Tools and Services," Global Software Engineering, 2009. ICGSE 2009. Fourth IEEE International Conference on, vol., no., pp.303-308, 13-16 July 2009.

Knob, F. F., "RiskFree4ppm: uma proposta de processo para o gerenciamento de portfólios de projetos distribuídos." Dissertação de Mestrado, PUCRS, 2007.

Karolak, D. W., "Global Software Development - Managing Virtual Teams and Environments." Los Alamitos, EUA: IEEE Computer Society, 159 p. 1998.

Bin X., "A Service Oriented Model for Role Based Global Cooperative Software Development," Convergence Information Technology, 2007. International Conference on , vol., no., pp.376-381, 21-23 Nov. 2007.

Bosch, J.; Bosch-Sijtsema, P., "Software Product Lines , Global Development and Ecosystems : Collaboration in Software Engineering," Collaborative Software Engineering, Springer Berlin Heidelberg, pp.77-92, 10 Mar. 2010.

Oshri I., Kotlarsky J. and Willcocks L.P., "Global Software Development: Exploring Socialization in Distributed Strategic Projects", Journal of Strategic Information Systems 16 (1), pp. 25-49. 2007.

Visser, C; "Connecting Time Zones and Languages in Follow-the-Sun: a routing model", Dissertação de Mestrado - Delf University of Technology – Holanda; 2009.

Gorton, Ian; Motwani, Sanjeev; "Issues in co-operative software engineering using globally distributed teams, Information and Software Technology", Volume 38, Issue 10, Pages 647-655, 1996.

Gupta, Amar; Mattarelli, Elisa; Satwik Seshasai, Joseph Broschak, "Use of collaborative technologies and knowledge sharing in co-located and distributed teams": Towards the 24-h knowledge factory, The Journal of Strategic Information Systems, Volume 18, Issue 3, Pages 147-161, 2009.