Um Framework de Recomendação para Alocação de Equipes de Desenvolvimento em Projetos Distribuídos de Linhas de Produto de Software

Vinicius S. dos Santos, Thaís A. B. Pereira, Bruno Luna Ribeiro, Glêdson Elias

COMPOSE – Component Oriented Software Engineering Group
Departamento de informática, Universidade Federal da Paraíba
João Pessoa, Paraíba, Brasil

{vinicius, thais, bruno}@compose.ufpb.br, gledson@di.ufpb.br

Abstract. *Software Product Line (SPL) and Distributed Software Development (DSD) approaches have been adopted by many companies to improve software quality, finding more qualified manpower, and reduce development costs and time. However, in such a scenario, teams allocation is not a trivial task, because it has to take into account properties of the software project, the teams and the context in which they are immersed. Thus, the framework proposed in this paper aims to provide recommendations on how to allocate teams to software components in distributed SPL projects by considering technical and non-technical aspects.*

Resumo. *Abordagens de Linhas de Produto de Software (LPS) e de Desenvolvimento Distribuído de Software (DDS) vêm sendo adotadas pelas empresas a fim de melhorar a qualidade do software, encontrar mão-de-obra mais qualificada, e reduzir custos e tempo de desenvolvimento. Contudo, a alocação de equipes de desenvolvimento num cenário desse tipo não é trivial, pois deve levar em consideração características do projeto de software, das equipes e do contexto onde as mesmas se encontram. Assim, o framework proposto neste artigo tem o objetivo de prover recomendações sobre como alocar equipes aos componentes de software em um projeto distribuído de LPS, considerando aspectos técnicos e não técnicos.*

1. Introdução

Nos últimos anos, a busca por melhores oportunidades de negócio, seja para reduzir custos de desenvolvimento ou alavancar vendas, bem como a escassez de profissionais qualificados, têm motivado a indústria de software a adotar abordagens de Desenvolvimento Distribuído de Software (DDS) [Gumm, 2006]. Nesse cenário caracterizado pela crescente complexidade dos produtos e do número de profissionais engajados, de acordo com [Paulish, 2003], Linhas de Produto de Software (LPS) contribuem para o planejamento e o gerenciamento de projetos distribuídos, uma vez que favorecem a definição de componentes de software bem estruturados, que podem ser paralelamente desenvolvidos por equipes geograficamente dispersas.

De fato, o desenvolvimento baseado em componentes promove a modularização da arquitetura, tornando o trabalho desempenhado pelas equipes de desenvolvimento menos dependente entre si [Ghezzi *et al.*, 2003]. Apesar disso, componentes mantêm certa dependência com outros componentes, ainda que estes sejam realmente bem projetados, já que devem ser integrados para as aplicações serem instanciadas, o que exerce influência sobre a necessidade de interação (ou comunicação) entre suas respectivas equipes de desenvolvimento [Sosa *et al.*, 2002]. Por sua vez, a comunicação

entre equipes geograficamente distribuídas pode não ser fácil, eficiente e efetiva, devido às inúmeras diferenças que podem existir entre elas, em termos de disponibilidade, recursos, conhecimento técnico e valores sócio-culturais [Herbsleb, 2007]. Assim, a qualidade dos produtos pode ser comprometida e o cronograma de atividades pode ser consideravelmente afetado, levando ao aumento de custos de desenvolvimento. Nesse contexto, estratégias para aprimorar a alocação de equipes às atividades de desenvolvimento de software exercem um papel de grande relevância.

Conforme observado em [Shen *et al.*, 2002], processos de alocação de equipes em projetos de software geralmente se baseiam na capacidade técnica das equipes (ou pessoas) e nas exigências das tarefas a serem desenvolvidas. Contudo, essa estratégia não é suficiente, pois como evidenciado, não somente a habilidade técnica das equipes influi na qualidade do trabalho por elas desempenhado, mas também a qualidade da comunicação entre equipes. Dessa forma, estratégias de alocação também devem buscar minimizar a necessidade de comunicação entre as equipes, bem como mitigar a dificuldade destas em estabelecer comunicação efetiva, uma vez que inevitavelmente sempre existirá comunicação. Visando atender essa necessidade, o presente artigo propõe um *framework* de recomendação para a alocação de equipes de desenvolvimento para a implementação de projetos distribuídos de LPS, que levam em consideração características do software, das equipes e do contexto onde as mesmas se encontram.

O restante do artigo está estruturado da seguinte maneira. A Seção 2 apresenta o *framework* proposto. A Seção 3 aborda os trabalhos relacionados, ressaltando as principais contribuições do *framework* proposto. Por fim, a Seção 4 apresenta as considerações finais e os trabalhos futuros.

2. Framework de Recomendação

O *framework* proposto visa oferecer suporte à tomada de decisão por parte do gerente de projeto sobre como alocar equipes de desenvolvimento para implementar os componentes de software de um projeto de desenvolvimento de LPS. Para tal, considerando o conjunto de equipes de desenvolvimento $E = \{e_1, e_2, ..., e_x\}$ que podem participar do projeto, o *framework* é constituído por quatro fases, conforme ilustrado na Figura 1, que geram três tipos de recomendações:

i. **Recomendação de módulos**: um conjunto de módulos de software $M = \{m_1, m_2, ..., m_n\}$ fracamente acoplados entre si, mas constituídos por componentes fortemente acoplados, que podem ser implementados de forma independente;

ii. **Recomendação de equipes**: um mapeamento $EM = \{(m, ET) | m \in M \wedge ET \subseteq E\}$ de cada módulo de software m para o subconjunto ET de equipes que são tecnicamente habilitadas a implementar cada módulo;

iii. **Recomendação de alocação**: um conjunto soluções de alocação $A = \{(m, e) | m \in M \wedge e \in ET\}$ de cada módulo de software m a uma equipe e pertencente ao conjunto de equipes tecnicamente habilitadas ET.

A fase de *análise da arquitetura* visa apoiar o engenheiro de software: (i) na definição do conjunto $M = \{m_1, m_2, ..., m_n\}$ de módulos fracamente acoplados, constituídos por componentes fortemente acoplados entre si; e (ii) na avaliação da dependência entre módulos, identificando a necessidade de comunicação entre equipes. Para tal, essa fase recebe como entrada as descrições estática e dinâmica da arquitetura da LPS, que pode ser tanto o projeto do domínio quanto de uma aplicação específica. Para identificar os módulos e suas dependências, esta fase adota um algoritmo de agrupamento, que explora a técnica de DSM (*Design Structure Matrix*) juntamente com

métricas arquiteturais do projeto. Uma DSM consiste numa matriz que representa os componentes do projeto e as dependências existentes entre eles [Yassine 2004].

Uma vez que os módulos foram identificados na primeira fase, a fase de *avaliação técnica* deve: (i) caracterizar os módulos em termos dos requisitos técnicos necessários para implementá-los; (ii) caracterizar as equipes em termos de conhecimentos técnicos requisitados para implementar os módulos; e, (iii) recomendar o mapeamento $EM = \{(m, ET) | \, m \in M \wedge ET \subseteq E\}$, associando cada módulo de software m ao subconjunto de equipes ET que são tecnicamente habilitadas para implementá-lo. Nesta fase, algoritmos, regras e políticas baseadas em lógica nebulosa (*fuzzy*) [Zadeah 1965] são adotados para avaliar os aspectos técnicos de módulos e equipes, e, assim, recomendar as equipes tecnicamente habilitadas para cada módulo.

Considerando que as dependências entre módulos e as equipes tecnicamente habilitadas já foram identificadas na primeira e segunda fase, respectivamente, a fase de *avaliação não-técnica* tem o propósito de: (i) caracterizar as equipes candidatas $E = \{e_1, e_2, ..., e_x\}$ segundo seus atributos não-técnicos, tais como localização, língua e disponibilidade, dentre outros atributos relevantes no contexto de DDS; (ii) identificar a necessidade de comunicação entre equipes segundo as *dependências entre módulos* e o *mapeamento entre módulos e equipes*; (iii) identificar a viabilidade de comunicação entre as equipes com base nos atributos não técnicos; e finalmente, (iv) gerar o conjunto de recomendações de alocação $A = \{(m, e) | m \in M \wedge e \in ET\}$, que atribui cada equipe específica e à um módulo de software m. Para tal, esta fase fundamenta-se na heurística de algoritmos genéticos [Holland 1975] para recomendar soluções aproximadas, resultando em uma abordagem computacionalmente menos dispendiosa quando comparada com algoritmos que buscam a solução ótima.

Após a alocação das equipes e a conclusão da implementação do projeto, a fase de *avaliação do desempenho das equipes* deve qualificar o desempenho das equipes alocadas, tanto em termos técnicos, verificando se o trabalho que lhes foi designado foi realizado adequadamente, quanto em termos não técnicos, avaliando como se sucedeu o relacionamento entre as equipes. Com base em tal avaliação, as descrições técnicas e não-técnicas das equipes são refinadas, melhorando a qualidade das recomendações geradas pelo *framework* em projetos posteriores.

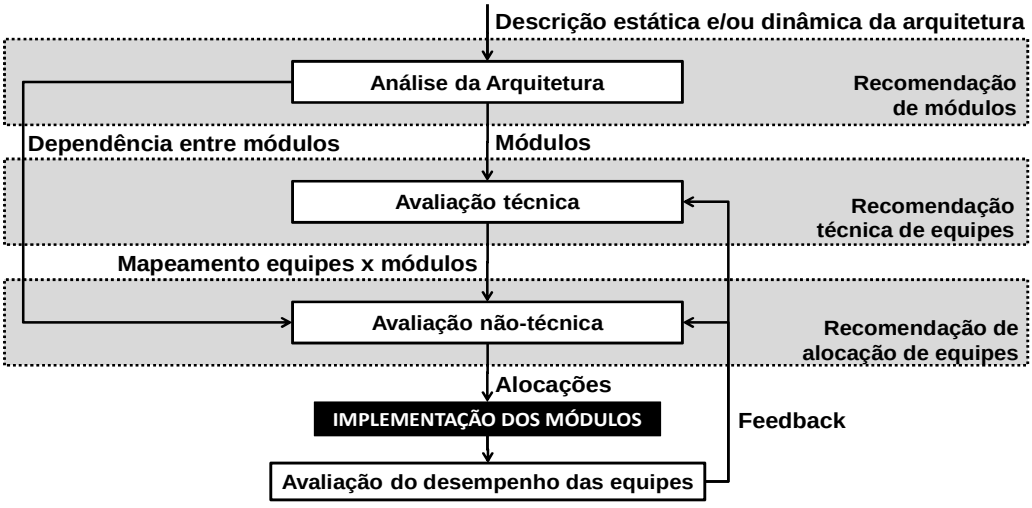

Figura 1. Visão geral do framework

2.1. Análise da arquitetura

Admitindo que dependências entre componentes de software exercem influência sobre a comunicação entre equipes de desenvolvimento, a fase de *análise da arquitetura* tem dois objetivos: identificar módulos fracamente acoplados, constituídos por componentes fortemente acoplados entre si; e medir o grau de dependência entre módulos.

A fim de atender a tais objetivos, nessa fase é empregada a técnica de DSM, que foi originalmente definida para apoiar o desenvolvimento de produtos na engenharia, como a indústria automotiva. A justificativa é que essa técnica facilita a visualização de dependências, bem como a realização de operações de agrupamento. Uma DSM consiste em uma matriz quadrada na qual as colunas revelam fontes de entrada para os componentes nas linhas e a existência de dependência entre pares de componentes é representada através de um símbolo ou valor binário (DSM binária), ou através de um valor numérico (DSM numérica), que consiste no peso da dependência.

Com base nisso, a fase de análise da arquitetura define métricas arquiteturais que devem ser utilizadas para mensurar o peso da dependência entre pares de componentes e assim construir a DSM do projeto (DSM numérica). Tais métricas estão baseadas na especificação estática e dinâmica da arquitetura, e expressam o acoplamento entre componentes sob diferentes perspectivas: a afinidade ou semelhança entre a funcionalidade oferecida pelos componentes, segundo as *features* que estes contemplam; a taxa de uso de um componente em termos de interfaces providas e requeridas e também de serviços providos e requeridos; o tipo de comunicação entre serviços (ex., síncrona ou assíncrona); e a intensidade de comunicação em termos do número de mensagens. A fim de que sejam consideradas as propriedades de variabilidade e opcionalidade de LPS, o uso das métricas deve ser feito de maneira coordenada, segundo as orientações também oferecidas nessa fase.

Além disso, esta fase define o algoritmo de agrupamento a ser empregado sobre a DSM para gerar módulos de componentes. Algoritmos de agrupamento para DSM em geral operam segundo a definição de uma função objetivo, que expressa o custo de uma configuração de componentes, e de uma estratégia de busca, que visa encontrar o menor valor possível para essa função. O algoritmo adotado consiste em uma adaptação do algoritmo de agrupamento de equipes, proposto em [Thebeau, 2001] para o contexto de componentes de software, através da reformulação da função de custo. A DSM agrupada resultante representa os módulos segundo os seus componentes constituintes e suas dependências internas (entre componentes em um mesmo módulo) e externas (entre componentes em módulos diferentes). Uma vez que os módulos são identificados, a dependência existente entre eles é mensurada, resultando no *modelo de dependência entre módulos*, que consiste em uma DSM cujas células representam a dependência entre pares de módulos de software, abstraindo os componentes internos aos módulos. O cálculo da dependência é realizado através de métricas similares àquelas usadas para medir a dependência entre componentes.

2.2. Avaliação técnica

A fase de *avaliação técnica* têm o objetivo de gerar um conjunto de recomendações de equipes que são tecnicamente hábeis para implementar os módulos de software identificados na primeira fase. Para alcançar tal objetivo, se faz necessário: descrever os módulos através dos requisitos técnicos necessários para sua implementação; descrever as equipes em termos de suas habilidades técnicas; e identificar as equipes mais capacitadas para implementar cada módulo com base na avaliação das informações destas e dos módulos.

A descrição dos módulos é feita com a criação do *modelo de descrição técnica dos módulos*. Esta atividade é realizada pelo arquiteto de software que descreve quais domínios e tecnologias (ferramentas, linguagens, etc.) estão envolvidas na implementação dos módulos, bem como os respectivos graus de conhecimento necessários para implementá-los. Por se tratar de uma LPS, também devem ser considerados aspectos sobre pontos de variação e componentes variantes, bem como identificar quais componentes do módulo fazem parte da arquitetura de referência e quais são de aplicações específicas.

Na descrição das equipes são utilizados formulários parar coletar informações sobre a experiência em projetos de DDS, bem como em projetos de LPS, englobando quantidade e complexidade dos projetos já realizados; conhecimento acerca do domínio dos projetos; habilidades tecnológica das equipes para atender aos requisitos definidos pelos módulos de software, tais como conhecimento em tecnologias, ferramentas, processos e certificações. A partir dessa descrição é gerado o *modelo de descrição técnica das equipes*. Como equipes se envolvem em vários projetos ao longo do tempo, esse modelo é atualizado cada vez que uma equipe se engaja em um projeto diferente.

Mensurar conhecimento ou experiência é uma tarefa subjetiva, uma vez que envolve aplicação de formulários, os quais são elaborados e respondidos segundo critérios humanos. Assim, um algoritmo de lógica nebulosa (*fuzzy*) é utilizado para lidar com tais imprecisões, já que o mesmo permite avaliar atributos por meio de termos com significado vago, como *baixo, médio* ou *alto* (termos difusos).

A avaliação técnica tem por base um conjunto de regras, representando políticas a serem adotadas para a seleção das equipes, que cruzam informações das equipes e módulos, produzindo uma métrica de adequabilidade técnica. Tais políticas são definidas numa matriz, onde as colunas representam os requisitos demandados pelos módulos e as linhas, as habilidades apresentadas pelas equipes. As células representam a adequabilidade técnica das equipes para implementar o módulo. Utilizando os termos difusos, o gerente de projeto pode alterar o conjunto de regras, refinando as políticas para atenderem às necessidades específicas de cada projeto. Por exemplo, se a política da organização é poupar equipes mais experientes para trabalhos mais difíceis, o processo de avaliação definirá como adequadas equipes que possuírem conhecimentos mais próximos aos demandados pelos módulos. Por outro lado, se o gerente deseja finalizar o projeto o mais rápido possível, o conjunto de regras é alterado de forma que equipes com qualificações mais altas sejam sempre mais adequadas.

2.3. Avaliação não-técnica

A fase de avaliação *não-técnica* é constituída por duas etapas, *descrição não-técnica das equipes* e *alocação de equipes*. A *descrição não-técnica* consiste na coleta de informações sobre o contexto em que se encontram as equipes, tais como os seus atributos geográfico, temporal, cultural e de reputação. Atributos geográficos representam a localização da equipe em diferentes perspectivas, como país, estado e cidade. Atributos temporais estão relacionados às horas de trabalho por dia e fuso horário, determinando segmentos de tempo de trabalho, que também são empregados para verificar a possibilidade de comunicação síncrona e assíncrona. Já os atributos culturais fazem referência aos idiomas que as equipes dominam e às características que representam costumes e padrões organizacionais de seus respectivos ambientes de trabalho. Por fim, a reputação especifica o quão boa é a interação entre equipes com base em trabalhos anteriores.

A etapa de *alocação das equipes* compreende a atividade de gerar recomendações sobre como alocar as equipes aos módulos de software, considerando a dependência existente entre os módulos *(modelo de dependência entre módulos)*, a habilidade técnica das equipes em desenvolver cada módulo *(modelo de mapeamento entre módulos e equipes)* e o contexto do relacionamento entre as equipes *(modelo de descrição não-técnica das equipes)*. Devido à grande quantidade de possíveis soluções, é usada a heurística de algoritmos genéticos. Dessa forma, inicialmente, as soluções são avaliadas com base nos atributos não-técnicos, sendo posteriormente melhoradas com operações de algoritmos genéticos, os quais também produzem mais soluções. O novo conjunto de solução é então avaliado com base no *modelo de dependência entre módulos*, a fim de que sejam comparadas as equipes que estão alocadas à módulos relacionados, realizando novas iterações até que o conjunto de soluções evolua e as mesmas sejam consideradas boas soluções de alocação.

Por exemplo, se módulos muito dependentes entre si são alocados às equipes *A* e *B*, é pressuposto que haverá maior necessidade de comunicação entre tais equipes. Assim, é preciso verificar a capacidade destas em estabelecer comunicação efetiva, conforme seus atributos não-técnicos. Se a equipe *A* só pode se comunicar em inglês e a equipe *B* apenas em espanhol, tais equipes provavelmente enfrentarão bastante dificuldade de comunicação, já que precisão aprender em curto prazo um novo idioma. No entanto, se a equipe *A* também conhece a língua espanhola, embora apenas em termos de leitura e escrita, as equipes podem estabelecer comunicação com relativa facilidade através de *chat* e *e-mail*. Nessa situação, a viabilidade da comunicação é então definida pela disponibilidade das equipes em termos de fuso-horário e horário de trabalho, já que *chat* exige que as equipes estejam conectadas no mesmo instante, ao contrário de *e-mail*. Uma vez geradas as recomendações, cabe ao gerente de projeto avaliá-las, modificá-las e escolher, com base em sua experiência e conhecimento, aquela mais adequada de acordo com os objetivos e características do projeto.

2.4. Avaliação do desempenho das equipes

A avaliação do desempenho das equipes tem o propósito de refinar as descrições técnica e não-técnica das equipes sempre que estas finalizam um projeto por intermédio do gerente do projeto, a fim de melhorar a qualidade das recomendações geradas pelo *framework*. Para tanto, essa etapa também introduz os atributos de *reputação técnica* e *de interação*, que são coletados através de questionários submetidos ao gerente de projeto e às equipes, respectivamente. A reputação técnica diz respeito à implementação dos módulos, e objetiva capturar aspectos como a qualidade final dos módulos implementados e o tempo gasto para desenvolvê-los. Já a reputação de interação diz respeito à satisfação das equipes ao se comunicarem com outras equipes em cada projeto, definida com base na dificuldade enfrentada por elas para estabelecer comunicação (devido às diferenças de horário e idioma, por exemplo).

3. Trabalhos correlatos

A alocação de equipes de desenvolvimento no contexto de DDS, e em particular de LPS, é algo que ainda é feito essencialmente com base na experiência do gerente do projeto, sendo escassas as abordagens que visam auxiliar nessa tarefa de forma realmente eficiente. Embora o *framework* proposto esteja focado na alocação de equipes geograficamente dispersas na atividade de implementação de uma LPS, aqui são considerados trabalhos de alocação de equipes no contexto de DDS em geral (ou numa perspectiva contrária, de alocação de atividades às equipes).

Em [Paulish, 2003] é apresentado um processo para desenvolvimento de LPS no qual somente a atividade de implementação é distribuída entre equipes geograficamente dispersas, que são coordenadas por uma equipe central. Nesse processo, a separação de unidades de implementação é feita na perspectiva de componentes de software, sendo definidas sobre eles algumas restrições em termos de tempo e esforço de desenvolvimento. A alocação das equipes aos componentes é feita considerando a habilidade técnica das equipes a cerca da funcionalidade dos componentes e as restrições definidas. Dessa forma, ao contrário do framework proposto, esse processo não considera as dependências entre componentes de software e os aspectos não técnicos das equipes, que são introduzidos pela separação geográfica.

Visando reduzir a necessidade de comunicação entre equipes de desenvolvimento, ocasionada por requisições de modificação de arquivos de implementação, em [Mockus e Weiss, 2001] é apresentado um algoritmo para encontrar a atribuição ótima de agrupamentos de arquivos em locais de desenvolvimento. Nesse modelo, o custo de uma configuração de agrupamento de arquivos é avaliado com base no fluxo de requisições de modificação entre locais. Contudo, não é discutida a alocação dos agrupamentos de arquivos aos locais, nem tampouco os atributos dos locais e dos arquivos que podem influir na quantidade de requisições de modificação, tornando o algoritmo insuficiente para ser adotado como base para definir boas soluções.

A fim de confrontar diferentes estratégias para alocação de tarefas em DDS, em [Setamanit *et al.*, 2007] foi desenvolvido um modelo que simula a colaboração entre locais e os efeitos da comunicação distribuída, segundo os critérios de diferença de fuso-horário, freqüência de contato, recursos disponíveis, diferenças culturais e distância. Apesar disso, o modelo não considera as dependências entre as tarefas, e também não oferece um algoritmo que busque encontrar a solução ótima de alocação.

Por fim, tal como o *framework* proposto, em [Lamersdorf e Munch, 2009] é apresentada uma ferramenta que gera um conjunto de sugestões de alocação de tarefas (projeto, implementação e teste) de desenvolvimento de software à equipes distribuídas geograficamente. Nessa ferramenta, as sugestões levam em conta o custo de se executar uma tarefa em um local, o custo de comunicação entre tarefas em locais diferentes com base em diferenças culturais e de fuso-horário entre as equipes e o acoplamento entre tarefas. Além disso, sugestões são ordenadas de acordo com a prioridade definida pelo gerente (custo, tempo de desenvolvimento ou qualidade). O trabalho, contudo, não descreve como são derivados os valores no qual as sugestões estão baseadas, o que torna impossível a completa avaliação do mesmo.

4. Considerações finais

Em projetos de desenvolvimento distribuído de LPS, a alocação de equipes de desenvolvimento aos componentes de software é uma tarefa complexa, geralmente deixada a cargo do gerente do projeto, para o qual falta o suporte adequado para analisar as inúmeras possibilidades existentes. Dessa forma, o presente artigo descreve uma abordagem sistemática de recomendar alocações de equipes, que prioriza a redução da necessidade de comunicação e das dificuldades geralmente enfrentadas para se comunicar efetivamente nesse cenário, e que por sua vez afetam a qualidade dos componentes resultantes.

Um aspecto bastante interessante do *framework* proposto é a independência entre as quatro fases que o compõe. Levando em consideração que os artefatos de entrada e saída das várias fases são claramente definidos, é possível a evolução ou até mesmo a mudança das técnicas e algoritmos empregados em qualquer uma delas, sem

introduzir qualquer impacto nas demais. Além disso, também é possível acrescentar novas fases, por exemplo, para tratar aspectos de negócio com o objetivo de minimizar os custos ou o tempo de desenvolvimento.

Ao adotar o conceito do *framework*, a abordagem pode ser vista como uma orientação estruturante do processo de alocação de equipes, que deve ser instanciada e personalizada para cada projeto de LPS no contexto de DDS. Nesta direção, a instanciação do *framework* em um projeto real de desenvolvimento de LPS está sendo iniciada para refinar e validar as várias fases e etapas do *framework*. Após a validação da proposta, também é planejado o desenvolvimento de ferramentas para automatizar as suas fases e etapas, bem como a introdução de uma fase para tratar aspectos de negócio, incluindo custo e tempo de desenvolvimento.

Agradecimentos. Este trabalho foi apoiado pelo Instituto Nacional de Ciência e Tecnologia para Engenharia de Software (INES)[1] e financiado pelo CNPq, processo 573964/2008-4.

Referências

Ghezzi, C. et al. (2003) “Fundamentals of Software Engineering”, Prentice Hall PTR.

Gumm, D.C. (2006) Distribution Dimensions in Software Development Projects: A Taxonomy, IEEE Software, 23(5), pp. 45-51.

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

Holland, J. H. (1975). “Adaptation in natural and artificial systems”. Ann Arbor, MI: The University of Michigan Press.

Lamersdorf, A., Munch, J. (2009) “TAMRI: A Tool for Supporting Task Distribution in Global Software Development Projects”, 4th IEEE international Conference on Global Software Engineering, pp. 322-327.

Mockus, A., Weiss, D. M. (2001) “Globalization by Chunking: A Quantitative Approach”. IEEE Software, 18(2).

Paulish, D.J. (2003) “Product Line Engineering for Global Development”, International Workshop on Product Line Engineering: The Early Steps.

Setamanit, S. et al. (2007) “Using simulation to evaluate global software development task allocation strategies”, Software Process: Improvement and Practice, 12(5), pp. 491-503.

Shen, M. et al (2002) “Multi-Criteria Task Assignment in Workflow Management Systems” 36th Annual Hawaii International Conference on System Sciences.

Sosa, M. E. et al. (2002) “Factors that influence Technical Communication in Distributed Product Development: An Empirical Study in the Telecommunications Industry”, IEEE Transactions on Engineering Management, 49(1), pp. 45-58.

Thebeau, R. E. (2001) “Knowledge management of system interfaces and interactions for product development processes”, Master of Science Thesis, MIT.

Yassine, A. (2004) "An Introduction to Modeling and Analyzing Complex Product Development Processes Using the Design Structure Matrix (DSM) Method", Quaderni di Management (Italian Management Review), www.quaderni-di-management.it, No.9, 2004.

Zadeh, L. A. (1965) “Fuzzy sets”, Information and Control, 8, pp. 338-353

[1] http://www.ines.org.br/