Gerência de Projetos e Processos de Desenvolvimento de Software: uma proposta de integração

Maurício Covolan Rosito[1], Daniel Antonio Callegari[1], Ricardo Melo Bastos[1]

[1] Faculdade de Informática – Pontifícia Universidade Católica do Rio Grande do Sul
Avenida Ipiranga, 6681 – 90619-900 – Porto Alegre – RS – Brasil

{mrosito, danielc, bastos}@inf.pucrs.br

Abstract. *Software organizations are constantly looking for better solutions when designing and using well-defined software processes for the development of their products and services. However, many existing software development processes lack for more project management skills in their models. This research proposes the definition of a model that integrates the concepts of project management and those available on the software development processes, building the steps towards a more comprehensive and automatable model. A prototype was developed in order to illustrate the proposed concepts.*

Resumo. *As organizações de software estão constantemente procurando por processos de software bem definidos para o desenvolvimento de seus produtos e serviços. Entretanto, muitos processos de desenvolvimento de software existentes apresentam carências no quesito de gerência de projetos. Esta pesquisa propõe a definição de um modelo que integra os conceitos de gerência de projetos e do processo de desenvolvimento de software formando uma base para a automatização do planejamento das atividades pertencentes a estas duas áreas de conhecimento. Um protótipo foi desenvolvido para demonstrar os conceitos propostos.*

1. Introdução

A crescente preocupação relativa ao desenvolvimento de software pode ser observada devido à adoção de práticas de engenharia de software pelas empresas [1]. O desenvolvimento de produtos de software requer o planejamento e a execução de atividades, definidas de acordo com o escopo do projeto, onde é necessário lidar tanto com assuntos gerenciais quanto técnicos. Em qualquer contexto deve-se considerar o fato de que projetos sempre são esforços únicos e temporários, além de envolverem um elevado nível de incerteza [2], [3].

A gerência de projetos (GP) em um ambiente de desenvolvimento de software é definida como a gerência das pessoas e de outros recursos por um gerente de projetos a fim de planejar, analisar, projetar, construir, testar e manter um sistema de informação [2], [3]. Para cumprir estes objetivos, um gerente de projetos precisa de algum tipo de suporte, geralmente baseado em uma metodologia de gerência de projetos, que permita lidar com diferentes variáveis de projeto, responsabilidades e tarefas.

Para este fim, existem diversas propostas na literatura ou práticas já realizadas nas empresas. Entretanto, conforme observado em [3], a maioria dos modelos ou guias voltados para a gerência de projetos, tal como o Project Management Body of Knowledge Guide (PMBOK) [4], não se dirigem especificamente a processos de desenvolvimento de software.

Os processos de desenvolvimento de software (PDSs), embora identifiquem a importância das atividades relacionadas à gestão, não tratam de forma adequada os aspectos relacionados à gerência de projetos. Tal fato é reforçado em [3] e [5], onde é destacado que dois dos mais importantes PDSs, respectivamente o Rational Unified Process (RUP) [6], por sua absorção no mercado, e o Object-oriented Process, Environment and Notation (OPEN) [7], por sua contribuição no meio acadêmico, necessitam de maior suporte no quesito de gerência de projetos. Tanto o RUP quanto o OPEN auxiliam na execução das melhores práticas para o desenvolvimento de software. No entanto, ambos os modelos mostraram-se incompletos em áreas essenciais de conhecimento da gerência de projetos, por exemplo, a gerência de aquisição, de comunicação e de pessoas. Cabe lembrar, também, que os PDSs atuais em geral não abordam de maneira adequada o tratamento das questões relacionadas a recursos humanos e outros tipos de recursos, tais como equipamentos e materiais envolvidos [3].

Portanto, se por um lado o PMBOK Guide pode fornecer uma perspectiva gerencial da solução, a visão sobre a produção deve ser obtida a partir de um modelo de processo de desenvolvimento de software, tal como os processos RUP e OPEN.

De acordo com [8], o processo de desenvolvimento de software é um dos principais mecanismos responsáveis por gerenciar e controlar os projetos e produtos de software. Durante o planejamento de atividades, o gerente de projetos pode necessitar interagir com outros departamentos da organização a fim de obter informações relevantes para o projeto (contatar o setor de recursos humanos sobre a necessidade de contratação de pessoal, por exemplo). Neste instante, há uma dissociação entre o fluxo de atividades de um projeto de software e os demais fluxos de atividades de suporte ao projeto da organização (aqui denominados **Fluxos Empresariais**). Conforme ilustrado na figura 1, ambos os fluxos de trabalho são executados de forma distinta e podem influenciar nos prazos das atividades e custos dos projetos. Também, pode haver uma relação de dependência entre as atividades pertencentes a estes dois tipos de fluxos de trabalho. Conseqüentemente, o gerente de projetos precisa de algum tipo de suporte que permita acompanhar o andamento das atividades destes fluxos de trabalho.

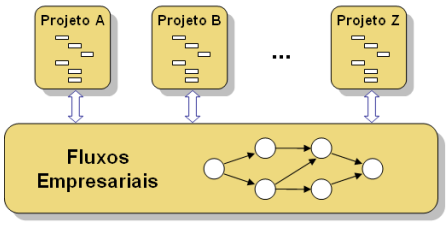


Figura 1. Relação entre os projetos e os Fluxos Empresariais

Este artigo apresenta uma proposta de modelo de integração entre a gerência de projetos e o processo de desenvolvimento de software denominado **Software Planning Integrated Model (SPIM)**. Este modelo inclui um conjunto de regras para o planejamento integrado de atividades gerenciais e produtivas para projetos de desenvolvimento de software. No SPIM são desenvolvidos os conceitos advindos da integração do PMBOK com o RUP em extensão à proposta apresentada em [3]. Com base no SPIM torna-se possível o desenvolvimento de ferramentas para o apoio ao processo de planejamento e acompanhamento das atividades gerenciais e produtivas para projetos de desenvolvimento de software. Neste sentido, foi desenvolvida a

ferramenta denominada **Software Planning Integrated Tool (SPIT)** que atua como um *Add-in* para uma ferramenta de gerenciamento de projetos já existente e muito utilizada.

Este artigo está organizado da seguinte forma: a Seção 2 discorre sobre o referencial teórico desta pesquisa; a Seção 3 apresenta os metamodelos base de integração; um modelo de integração entre a GP e o PDS é apresentado na Seção 4. Finalmente, na Seção 5, são apresentadas as conclusões e as próximas etapas.

2. Referencial Teórico

2.1. Project Management Body of Knowledge – PMBOK

Reconhecido internacionalmente pelo seu esforço em definir normas e dar suporte aos profissionais de gerência de projetos, o Project Management Institute (PMI) [9] publicou um guia geral de gerência de projetos, o PMBOK Guide. O Project Management Body of Knowledge reúne as melhores práticas aplicáveis à maioria dos projetos e sobre as quais há um amplo consenso sobre o seu valor e utilidade.

De acordo com [4] e [9], o principal objetivo do PMBOK Guide é identificar um subconjunto dos conhecimentos sobre gerência de projetos que seja reconhecido, genericamente, como sendo uma coleção de boas práticas. O PMBOK, porém, não é um processo sem seu sentido estrito, pois não determina quais são as ações e nem indica como estas devem ser executadas para o correto desenvolvimento de um projeto. Além disso, o PMBOK é dito mais compatível com atividades industriais e, portanto, não aborda especificamente processos de desenvolvimento de software [3].

Para efeito da discussão proposta por este artigo, observa-se que os conceitos do PMBOK são, em sua maioria, representados por textos descritivos. Entretanto, objetivando comparar dois modelos e, posteriormente, executar a integração entre eles, deve-se representá-los sob estruturas compatíveis – neste artigo será utilizado o metamodelo previamente desenvolvido em [3], usando notação UML (figura 2).

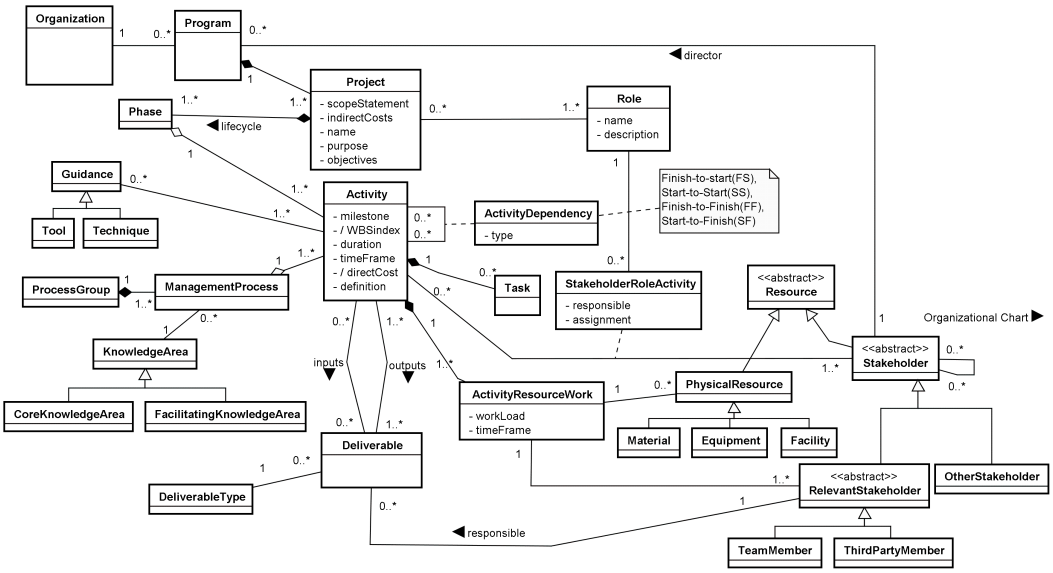


Figura 2. Metamodelo de gerência de projetos baseado no PMBOK Guide [3]

2.2. Rational Unified Process - RUP

O RUP é um processo iterativo de desenvolvimento de software desenvolvido pela empresa IBM Rational Software, originado a partir do metamodelo SPEM [10], [11]. O RUP atua como um *framework* que pode ser adaptado e estendido de acordo com as características do processo de desenvolvimento de software da organização [12]. Segundo [6], o RUP apresenta um modelo semântico, ilustrado na figura 3, contendo seus principais elementos e relacionamentos.

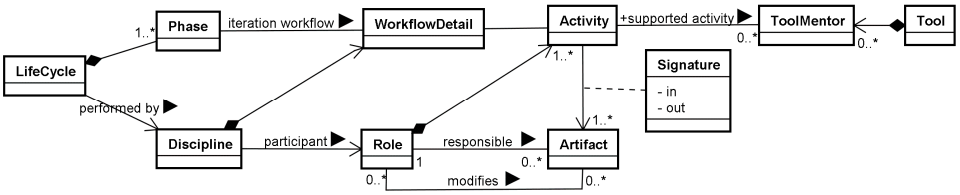


Figura 3. Modelo semântico do RUP [6]

Na figura 3, a classe **Lifecycle** representa o ciclo de vida de desenvolvimento de um software. Este conceito é particionado em um conjunto de quatro fases (classe **Phase**). A classe **Discipline** divide os elementos de processo em áreas de interesse. Um papel (classe **Role)** representa o elemento responsável por desempenhar atividades (**Activity**) para produzir ou modificar os artefatos (**Artifact**) do processo. As informações de como os papéis devem colaborar entre si através de suas atividades são definidos pela classe **Workflow Detail**. A classe **Artifact** descreve os tipos de produtos de trabalho que são produzidos ou consumidos no desempenho de atividades. Assim, a classe associativa **Signature** indica que um artefato é utilizado como entrada ou saída de uma atividade. A classe **Tool** descreve as ferramentas que podem ser utilizadas auxiliando a produção ou modificação de um artefato. Finalmente, a classe **ToolMentor** descreve o uso de ferramentas no contexto de algumas atividades.

3. Metamodelos Base de Integração

O estudo detalhado dos modelos do PMBOK e RUP, reforçado posteriormente com a inclusão do OPEN para seu refinamento, permitiu identificar como são organizados e quais as relações válidas entre os elementos de cada modelo. Através da integração entre a gerência de projetos e os processos de desenvolvimento de software foi possível identificar as principais características e discrepâncias entre os elementos de tais modelos. Conforme citado anteriormente, o metamodelo de referência do PMBOK [3] inclui os elementos necessários para a gerência de projetos, enquanto que os conceitos de processos de desenvolvimento de software são obtidos pelo RUP ou pelo OPEN.

O critério de integração entre os modelos seguiu o conjunto de regras identificado por [3], que afirma que ao se realizar uma integração entre dois modelos, as seguintes situações podem ocorrer: (a) Uma sobreposição de conceitos (duas classes com o mesmo conceito em cada modelo): neste caso, pode-se transformar e unir estas duas classes em um único conceito dentro de um pacote comum; (b) Uma Relação entre conceitos (uma classe de um dos modelos se relaciona com alguma outra classe de outro modelo, mas estas classes não representam exatamente o mesmo conceito): deve-se manter as classes em seus modelos originais e criar uma associação entre elas; e (c) Conceitos independentes (classes com conceitos independentes e distintos): deve-se deixar cada classe em seu próprio pacote.

A proposta de integração entre gerência de projetos e processos de desenvolvimento de software apresentado neste trabalho é constituída de três pacotes: um para os conceitos de GP, outro para os relacionados aos PDSs e, finalmente, um pacote comum ("Common") que une os conceitos que ocorrem em ambos os modelos.

As seções a seguir apresentam uma discussão mais elaborada, que envolve o desenvolvimento dos dois metamodelos de integração desenvolvidos nesta pesquisa.

3.1. Metamodelo Integrado entre o PMBOK e o RUP

De modo a explicar o metamodelo de integração PMBOK+RUP (figura 4), seus principais elementos serão descritos. Devido às limitações de espaço, serão apresentadas as principais extensões realizadas para o desenvolvimento do SPIM ao metamodelo integrado entre o PMBOK e o RUP proposto em [3].

É importante ressaltar que alguns conceitos relacionados à gerência de projetos que estão contidos nos processos de desenvolvimento de software (neste caso, no RUP e no OPEN) foram propositadamente deslocados para o pacote de classes gerenciais (pacote PMBOK) com o objetivo de deixar mais explícita a classificação dos conceitos de GP e do processo de desenvolvimento de software. Também optou-se por manter os conceitos na língua inglesa para facilitar a comparação com os modelos originais.

Neste modelo, a classe **Organization** representa uma empresa que se organiza por programas (classe **Program**). Os programas são grupos de projetos (classe **Project**) designados a alcançar um objetivo estratégico. As organizações geralmente dividem os projetos em várias fases (classe **Phase**) visando um melhor controle gerencial.

Os recursos necessários para um projeto são explicitamente descritos no sub-pacote **Resources**. Sendo assim, pessoas, equipamentos e locais são representados pela classe **Resource**. Estes recursos são divididos em recursos ativos (classe **Stakeholder**) e não-ativos (classe **PhysicalResource**). Stakeholders correspondem às pessoas e organizações cujos interesses são afetados pelo projeto [4]. A classe associativa **ProjectStakeholder** permite definir se um stakeholder é "chave" para o projeto [4]. O sub-pacote **Resources** também contém informações sobre a disponibilidade de cada recurso ao atribuí-lo às atividades, mesmo que realizado de forma manual ou automática, através da classe **ResourceAvailability** [3]. Esta classe permite automatizar os processos de alocação de recursos em projetos de software. Paralelamente, a definição da carga de trabalho (atributo *workload*) dos recursos físicos (ao associar-se a diferentes atividades) é representada pela classe **ActivityPhysicalResourceWork**.

Uma atividade (classe **Activity**) pode ser classificada como atividade produtiva (**ProductiveActivity**) ou atividade gerencial (**ManagerialActivity**). As atividades produtivas são diretamente relacionadas com a construção do software. Entretanto, as atividades gerenciais podem pertencer ao fluxo de desenvolvimento de software (*isExternal=false*) ou aos fluxos empresariais da organização (*isExternal=true*). Logo, identificam-se três tipos de atividades: **produtivas**, **gerenciais** e **gerenciais de apoio**. Cada atividade pode pertencer a um ou mais *baselines*. Em cada geração da *baseline*, uma atividade deve manter os relacionamentos com os papéis e produtos de trabalho. Assim, objetivando-se manter estes relacionamentos das atividades com outras entidades do modelo durante a geração de *baselines*, decidiu-se assumir que a classe **Activity** conterá uma agregação de uma ou mais classes **ActivityDetail**.

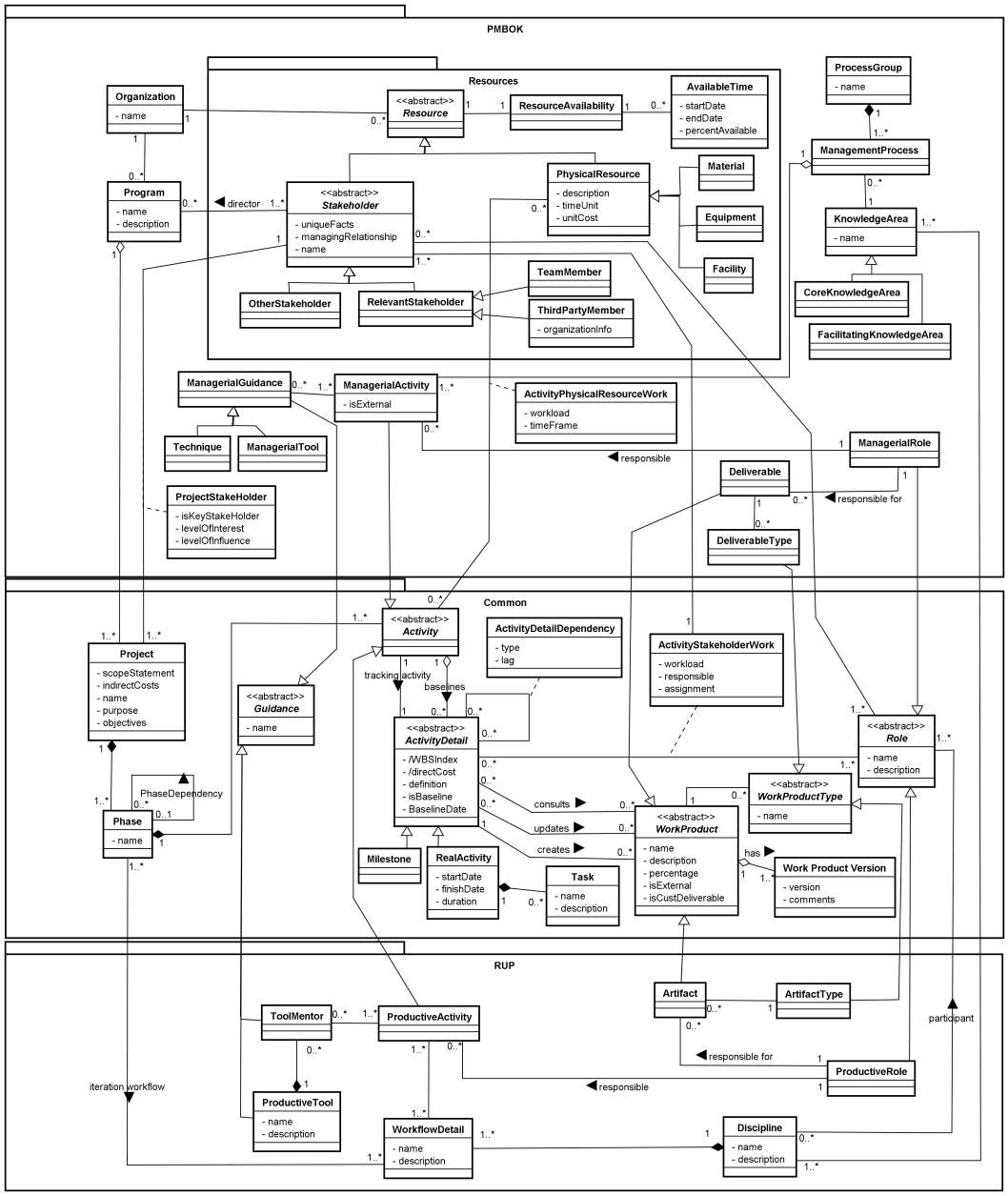


Figura 4. Metamodelo de integração PMBOK+RUP

Os stakeholders podem desempenhar diversos papéis (classe **Role**) durante a execução das atividades do projeto. Assim, para cada associação entre um papel e uma atividade (representado pela classe associativa **ActivityStakeholderWork**) deve haver também uma associação dessa atividade com um stakeholder capaz de desempenhar aquele papel. Além disso, como o conceito de papéis (classe **Role**) aparece em ambos os modelos, estes foram divididos em papéis gerenciais (classe **ManagerialRole**) e papéis produtivos (classe **ProductiveRole**), tal como ocorreu com as atividades.

Um produto de trabalho, por sua vez, (classe **WorkProduct**) pode ser classificado como um produto gerencial (classe **Deliverable**) ou produtivo (classe

Artifact), o qual deve estar associado a um tipo de produto (classes **DeliverableType** e **ArtifactType**, respectivamente). Cabe salientar que o modelo faz distinção das possíveis relações entre uma atividade e um artefato (criar/atualizar/consultar). Isto é importante para garantir a consistência do modelo a partir de regras, como a de número 16 na tabela apresentada mais adiante na seção 4.

Em relação à gerência de projetos, o PMBOK Guide representa suas práticas em duas dimensões lógicas. Uma dimensão descreve as áreas de conhecimento (classe **KnowledgeArea**) enquanto que a outra dimensão descreve os processos gerenciais de um projeto (classe **ManagementProcess**), os quais estão contidos em cinco grupos de processo (classe **ProcessGroup**). As áreas de conhecimento são responsáveis por descrever as principais competências que os gerentes de projeto devem desenvolver e derivam as áreas centrais (classe **CoreKnowledgeArea**) e as de apoio (classe **FacilitatingKnowledgeArea**). Assim, cada atividade gerencial pertence a um processo gerencial, sendo também relacionada a uma área de conhecimento.

Em contrapartida, o RUP define uma disciplina (classe **Discipline**) como sendo a divisão de elementos de processo em áreas de interesse. Cada disciplina é composta por um ou mais fluxos de trabalho (classe **WorkflowDetail**). Os fluxos de trabalho definem como os papéis produtivos devem colaborar entre si através de suas atividades.

3.2. Metamodelo Integrado entre o PMBOK e o OPEN

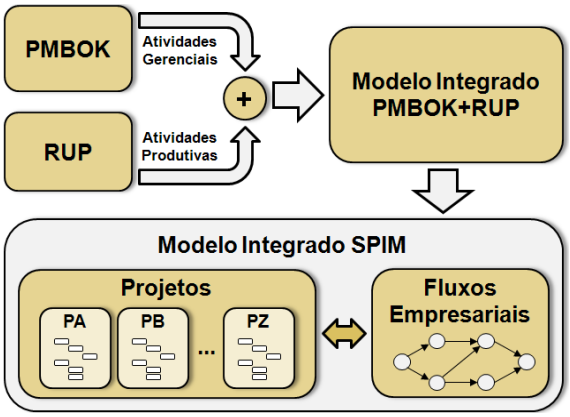
O metamodelo de integração PMBOK+OPEN desenvolvido apresenta uma estrutura similar ao apresentado na seção anterior, substituindo apenas o pacote referente ao processo de desenvolvimento de software. Assim, as classes dos pacotes PMBOK e Common são as mesmas que as apresentadas no metamodelo de integração PMBOK+RUP. Também permanecem inalterados os relacionamentos entre as classes pertencentes a estes dois pacotes. Os dois PDSs, porém, possuem características particulares que são refletidas em classes distintas e em diferentes relacionamentos com os pacotes PMBOK e Common. Embora não apresentada em detalhes aqui por questões de espaço, a integração com o OPEN permitiu tanto a confirmação quanto a adequação dos conceitos propostos no modelo final.

Para demonstrar a viabilidade dos conceitos propostos, foi desenvolvido o modelo integrado SPIM, que auxilia o planejamento de projetos considerando os elementos que compõem o processo de desenvolvimento de software. A atual versão do modelo implementa exclusivamente os conceitos advindos da integração do PMBOK com o RUP. Uma implementação que aborda também o OPEN está prevista futuramente.

4. Modelo de Integração entre a Gerência de Projetos e o Processo de Desenvolvimento de Software

O estudo da integração com os dois processos distintos de desenvolvimento de software (RUP e OPEN) permitiu elaborar uma visão mais ampla de como a gerência de projetos pode positivamente contribuir no desenvolvimento de um produto de software. Assim, os metamodelos definidos durante esta pesquisa (PMBOK+RUP e PMBOK+OPEN) fornecem a estrutura conceitual necessária para o desenvolvimento de um modelo que auxilie o planejamento de projetos considerando os elementos que compõem o PDS.

Conforme salientado anteriormente, o modelo de integração entre a GP e o PDS desenvolvido nesta pesquisa, denominado **Software Planning Integrated Model (SPIM)**, representa os conceitos referentes à integração do PMBOK com o RUP. A figura 5 representa as etapas que constituíram o desenvolvimento deste modelo de integração.



O estudo detalhado dos modelos do PMBOK e RUP permitiu identificar características relevantes da GP e dos PDSs. Este estudo resultou no desenvolvimento de um modelo de integração PMBOK+RUP. Posteriormente, o modelo SPIM foi desenvolvido implementando os conceitos advindos desta integração.

Também identifica-se a dissociação entre o fluxo de atividades produtivas de um projeto e os demais fluxos empresariais da organização. Assim, o modelo SPIM tem como objetivo apoiar o processo decisório de uma organização permitindo o planejamento integrado de atividades gerenciais e produtivas.

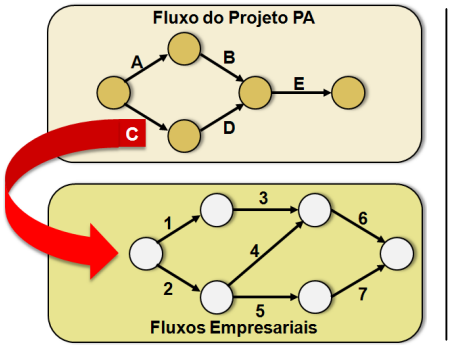
Figura 5. Etapas do desenvolvimento do modelo de integração SPIM

Conforme mencionado anteriormente, como o gerente de projetos pode não possuir todas as informações relevantes para o projeto, poderão ocorrer interações com outros departamentos da organização durante o planejamento de atividades de um projeto de software. Assim, o fluxo de atividades de um projeto de software pode interagir com os demais fluxos de atividades da organização (**Fluxos Empresariais**). É importante salientar que ambos os fluxos de trabalho são executados em paralelo, possuem recursos próprios e podem influenciar nos prazos das atividades e custos do projeto de software. As ferramentas atuais mais utilizadas, porém, não apresentam uma solução que permita o planejamento integrado de atividades gerenciais e de produção – e esta carência pode resultar em distorções no planejamento de projetos pela desconsideração das dependências entre as atividades dos diferentes fluxos de trabalho.

Durante o planejamento de atividades de um novo projeto de software, por exemplo, o gerente de projetos informa ao setor de recursos humanos sobre a necessidade de contratação de um administrador de banco de dados. Neste caso, constata-se a existência de uma relação de dependência entre as atividades do projeto de software (tais como, a modelagem do banco de dados) com as atividades pertencentes ao fluxo de trabalho do setor de recursos humanos referentes à contratação do profissional requerido para executar a atividade de produção. A dificuldade para visualizar esta interdependência dos fluxos de trabalho durante o planejamento de atividades gerenciais e de produção pode afetar negativamente o projeto, resultando, por exemplo, no aumento dos custos e em atrasos nos prazos do projeto. A preparação técnica e a liberação de uma sala ou equipamento testes são outros exemplos cujas atividades não são exclusivas de um projeto em especial, mas de um fluxo comum da empresa, compartilhado pelos projetos em andamento e que utiliza recursos não alocados diretamente ao projeto de software.

A solução proposta nesta pesquisa permite que cada instância de um **fluxo empresarial** seja registrada como correspondente a uma atividade gerencial de apoio ao projeto de desenvolvimento de software (figura 6). Desta forma, cada atividade gerencial de apoio pode ter claramente definida as suas relações de dependência com

outras atividades do projeto. Isto permite a integração com uma ferramenta de *workflow*, de maneira que este último seja responsável por atualizar os prazos das atividades de apoio ao projeto (um fluxo empresarial pode atender múltiplos projetos simultaneamente; daí a necessidade de um mecanismo tal como um *workflow*.)



Neste exemplo, a atividade "C" do projeto PA representa a execução total de um fluxo de trabalho empresarial (seus atributos, tal como o tempo, dependem diretamente dos atributos das atividades do fluxo empresarial). Assim, esta atividade do projeto só é dada como finalizada depois que todas as atividades do fluxo empresarial tiverem sido executadas. Assim, tão logo as atividades "6" e "7" do *workflow* terminem, a atividade "C" do projeto é dada como finalizada, liberando a execução para a atividade "D". Cabe salientar que os exemplos estão na notação de redes PERT [13].

Figura 6. Inter-relacionamento dos fluxos de trabalho

A integração dos modelos também permitiu que um conjunto de regras (ou restrições) pudessem ser derivadas (Tabela 1). Tais características, em sua maioria para garantir a consistência do modelo, não puderam ser expressas no diagrama devido à falta de expressividade do diagrama de classes da UML para este fim. Estas regras podem, contudo, ser definidas através de uma linguagem para especificação formal de restrições, tal como a Object Constraint Language (OCL) [14]. Neste artigo optou-se por grafá-las em linguagem natural, para facilitar a compreensão.

Tabela 1. Conjunto de Restrições Adicionais para o Modelo Integrado

	Restrições Adicionais ao Modelo Integrado
1.	Um programa deve possuir um diretor. Logo, um stakeholder que é diretor de um programa deve possuir um papel gerencial;
2.	Um projeto deve ter apenas um stakeholder-chave, ou seja, um recurso responsável por direcionar e fundamentar o projeto (atributo *isKeyStakeholder* de **ProjectStakeholder**);
3.	Uma fase não pode ter ela mesma como predecessora ou antecessora;
4.	As fases do projeto não podem ocorrer em paralelo, de maneira que uma fase deve ser totalmente finalizada antes de iniciar outra;
5.	Uma atividade não pode ter ela mesma como predecessora ou antecessora;
6.	O fluxo de atividades não pode resultar em um ciclo; por exemplo, a atividade A é pré-requisito para a atividade B e a atividade B é pré-requisito para a atividade A;
7.	Uma mesma atividade não pode criar, modificar ou consultar um mesmo artefato. Para realizar tais operações devem ser criadas atividades distintas;
8.	Uma atividade gerencial deve ter pelo menos um papel gerencial como um de seus papéis;
9.	Uma atividade produtiva deve ter pelo menos um papel produtivo como um de seus papéis;
10.	O stakeholder responsável por uma atividade gerencial deve possuir um papel gerencial;
11.	O stakeholder responsável por uma atividade produtiva deve possuir um papel produtivo;
12.	Cada atividade do projeto deve ser de responsabilidade de apenas um indivíduo, mesmo que muitas pessoas venham a trabalhar naquela atividade;
13.	Uma atividade gerencial não pode produzir ou modificar um produto de trabalho produtivo, somente um produto de trabalho gerencial. Porém, esta atividade pode consultar um produto de trabalho produtivo;
14.	Uma atividade produtiva não pode produzir ou modificar um produto de trabalho gerencial, somente um produto de trabalho produtivo. Porém, esta atividade pode consultar um produto de trabalho gerencial;

15.	Para cada associação entre um papel e uma atividade (representado pela classe **ActivityStakeholderWork**) deve haver também uma associação dessa atividade com um stakeholder capaz de desempenhar aquele papel;
16.	Uma atividade somente pode atualizar ou consultar um produto de trabalho que já tenha sido criado por uma atividade antecessora;
17.	Uma atividade pode ou não ter *baselines*. Se tiver, a atividade original deve ter o atributo *IsBaseline=false* e todas as outras atividades (relacionadas via a associação *Baselines*) devem manter *IsBaseline* como *true*.
18.	Na classe associativa **ActivityStakeholderWork**, o stakeholder associado deve possuir o papel que se está associando à atividade.
19.	O papel do stakeholder envolvido deve ser compatível com o tipo de atividade (gerencial ou produtiva).

O protótipo, denominado **Software Planning Integrated Tool (SPIT)**, foi desenvolvido para demonstrar os conceitos propostos pelo modelo integrado SPIM e pelo conjunto de regras descritas acima, fornecendo, desta forma, o suporte necessário para trabalhar com diferentes fluxos de trabalho, variáveis de projeto, responsabilidades e tarefas. O SPIT (figura 7), foi desenvolvido na linguagem C#.Net e atua como um *Add-in* para a ferramenta de gerenciamento de projetos Microsoft Office Project [21]. As informações necessárias para as validações dos conceitos propostos pelo modelo integrado SPIM estão contidas em campos customizados desta ferramenta comercial.

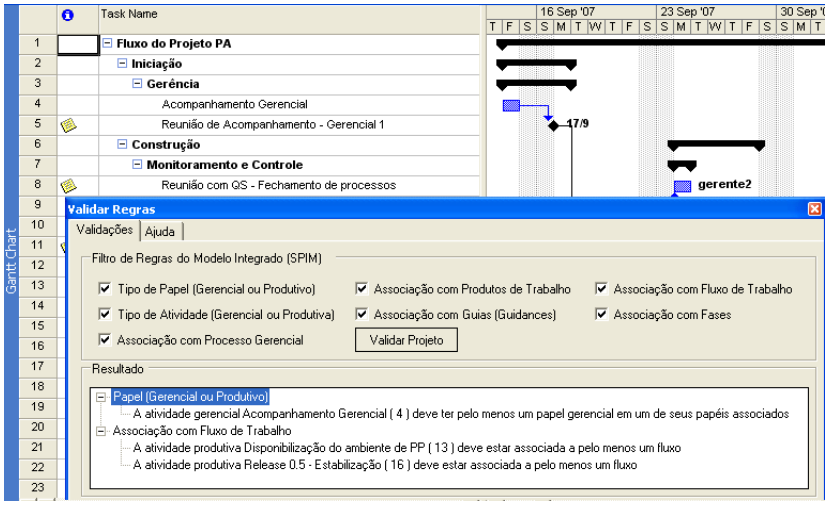


Figura 7. Protótipo Software Planning Integrated Tool (SPIT)

5. Considerações Finais

Este trabalho apresentou uma proposta para a integração dos principais conceitos sobre gerência de projetos e processos de desenvolvimento de software. Inicialmente, identificou-se a importância das atividades de gerência de projetos durante um projeto de desenvolvimento de software. Em seguida, observou-se a carência existente no quesito de GP nos PDSs atuais. Após uma análise individual de cada metamodelo base, foram apresentadas as principais colaborações ao metamodelo proposto em [3]. Finalmente, foi apresentado o protótipo que implementa o modelo de integração SPIM e o seu conjunto de restrições.

Segundo estudos empíricos, a eficácia de uma organização depende, em boa parte, do sucesso de seus projetos [15], [16]. Desta forma, muitos pesquisadores

trabalham na investigação dos fatores de sucesso dos projetos, tais como a definição de produto, qualidade de execução e técnicas de gerência de projetos [16], [17].

A importância da utilização de um PDS padrão deve-se ao fato de que este atua como um guia para a execução de todos os projetos dentro de uma organização. Assim, muitos processos tais como o RUP, o Extreme Programming (XP) [18], o Microsoft Solutions Framework (MSF) [19] e o OPEN estão sendo utilizados como base no desenvolvimento de processos de desenvolvimento de software padrão.

A análise de como o conhecimento sobre gerência de projetos permite aperfeiçoar os processos de desenvolvimento de software atuais torna possível o desenvolvimento de novas ferramentas capazes de suportar diferentes níveis de automatização no planejamento e na execução de atividades num projeto de software. Por essa razão, o modelo integrado foi desenvolvido com os seguintes os objetivos: (a) permitir o planejamento de projetos considerando os elementos que compõem a GP e o PDS; (b) fazer a distinção dos tipos de atividades (gerencial ou produtiva) e produtos; (c) adicionar a noção de disponibilidade a um recurso, de modo que permita automatizar os processos de alocação de recursos em projetos de software; (d) prever a informação de carga de trabalho (*workload*) para as associações de um papel, atividade e stakeholder, preparando para a automatização; e (e) distinguir as possíveis relações entre uma atividade e um artefato (criar/atualizar/consultar);

A falta de uma metodologia de gerenciamento de projetos, bem como o crescimento da complexidade e do volume dos projetos em uma organização, contribui para o aumento das dificuldades relacionadas ao desenvolvimento de projetos [3], [1]. Além disso, durante o planejamento de atividades, pode haver interações com outros departamentos da organização, pois muitas vezes o gerente de projetos não detém todas as informações relevantes para o projeto. O protótipo desenvolvido implementa um conjunto de regras objetivando fornecer suporte ao gerente de projetos para lidar com diferentes fluxos de trabalho, variáveis de projeto, responsabilidades e tarefas. Mais uma vez, o objetivo é facilitar o trabalho do gerente de projetos, que tem de levar em conta cada vez mais elementos em suas decisões e com cada vez menos tempo.

Durante o desenvolvimento desta pesquisa foram identificadas algumas questões que necessitam de maior desenvolvimento: (a) formalização das restrições do metamodelo através da linguagem OCL; (b) ampliação do estudo sobre a integração do processo de gerência com outros processos de desenvolvimento de software, tais como XP e MSF; e (c) desenvolvimento de um protocolo para a avaliação do modelo proposto com empresas de software, utilizando o protótipo em projetos reais.

Acredita-se ser possível estender estes metamodelos de integração com outros PDSs, porque, de acordo com [20], os diferentes modelos de desenvolvimento de software compartilham atividades fundamentais, tais como a especificação, projeto, implementação, validação e evolução do software. Atualmente, existem diversos artigos na literatura sobre diferentes integrações da GP com os PDSs, tais como [3] e [22]. Entretanto, parece não haver estudos suficientes para suprir a carência no quesito de gerência de projetos destes processos. Dessa forma, a continuidade desta pesquisa indica novas contribuições para engenharia de software, melhorando nossa compreensão dos relacionamentos da GP com os PDSs. Em um trabalho paralelo, um modelo multi-critérios para alocação de recursos já está sendo integrado ao protótipo.

Referências

[1] R. Pressman, "Software Engineering : a practitioner's approach", 5ª Ed., McGraw-Hill, 2001.

[2] K. Schwalbe, "Information Technology Project Management", 2ª Ed., Thomson Learning, Canada, 2002.

[3] D. Callegari, e R. Bastos, "Project Management and Software Development Processes: Integrating RUP and PMBOK", ICSEM - International Conference on Systems Engineering and Modeling, Haifa, Israel, 2007.

[4] PMBOK – "A Guide to the Project Management Body of Knowledge", 2000.

[5] B. Henderson-Sellers, R. Dué, I. Graham, e G. Collins, "Third generation OO processes: a critique of RUP and OPEN from a project management perspective", ASPEC - Seventh Asia-Pacific Software Engineering Conference, 2000.

[6] PEP – "A RUP Adoption Process. Rational Process Engineering Process", Rational Unified Process, 2003.

[7] I. Graham, B. Henderson-Sellers, e H. Younessi, "The OPEN Process Specification", Addison Wesley, 1997.

[8] W. S. Humphrey, T.R. Snyder, and R.R. Willis, "Software Process Improvement at Hughes Aircraft", Institute of Electrical and Electronic Engineers, pp. 11-23, 1991.

[9] PMI Web Site, http://www.pmi.org/info/default.asp, Novembro, 2006.

[10] Jacobson, I., Booch G., Rumbaugh J., "The Unified Software Development Process", Upper Saddle River, Addison Wesley, 2001.

[11] A. Bencomo, "Extending the RUP, Part 1: Process Modeling", http://www-128.ibm.com/developerworks/rational/library/4721.htm, Novembro, 2006.

[12] Rational Software Corporation Web Site, "Rational Unified Process: Best Practices for Software Development Teams", http://www.rational.com/products/rup/whitepapers.jsp, Novembro, 2006.

[13] R. Burke, "Project Management: Planning and Control Techniques". 3ª Ed., John Wiley & Sons, 2001.

[14] J. Warmer, A. Kleppe, "The Object Constraint Language - Precise Modeling with UML", Addison Wesley, 1999.

[15] H. Kerzner, "Applied project management: best practices on implementation", John Wiley & Sons, 2000.

[16] R.G. Cooper, "Winning at New Products: Accelerating the Process from Idea to Launch", Addison-Wesley, 1993.

[17] J. Pinto, and D. Slevin, "Critical factors in successful project implementation", IEEE Transactions on Engineering Management, pp.22–7, 1987.

[18] K. Beck, "Extreme Programming Explained: Embrace Change", Ad. Wesley, 1999.

[19] R. Hundhausen, "Working with Microsoft Visual Studio 2005 Team System", White Paper, 2006.

[20] I. Sommerville, "Software engineering", 5ª Ed., Addison-Wesley, 1995.

[21] Microsoft Project Web Site, www.microsoft.com/Project/, Março, 2007.

[22] S. S. Alhir, "Integrating the Project Management Body of Knowledge (PMBOK) Guide and the Unified Process (UP)", White Paper, 2003.