Comparação de Metamodelos de Processos de Desenvolvimento de Software

Paula Ventura Martins, Alberto Rodrigues Silva

Abstract — Sendo o metamodelo *Software Process Engineering Metamodel* (SPEM) promovido no âmbito da OMG como norma para especificação de processos de desenvolvimento de software, é essencial a sua comparação com os processos existentes. Este artigo analisa a expressividade e adequabilidade dos conceitos de suporte, de componentes e do ciclo de vida do processo segundo a terminologia SPEM, através de uma comparação com três processos, designadamente RUP, XP e MSF. Em particular, a definição de um conjunto de padrões observados no contexto dos processos de desenvolvimento de software, permitirá capturar elementos comuns à maior parte dos processos comerciais.

Termos — Metamodelos, Padrões, Modelos de processos, Normas.

1 Introdução

O desenvolvimento de software enfrenta enormes pressões, sendo reconhecido que a gestão efectiva de processos de desenvolvimento de software é um factor chave para o seu sucesso [1]. A importância dos processos de desenvolvimento de software (de ora em diante designado em geral por "processo") deriva do facto de permitir melhorar as previsões, melhorar a qualidade, minimizar custos e tempos na execução do projecto, assistir os interessados, fornecer estrutura para melhorias futuras [2]. Porém, a implementação de processos não se revela uma tarefa fácil [1]: (1) os processos são complexos, i. e., altamente iterativos, com paralelismo de tarefas e relacionamentos não lineares; (2) o ambiente é instável pois ocorrem mudanças rápidas da tecnologia, mudanças nos requisitos de negócio, elevados índices de rotatividade dos recursos humanos ou pressões de mercado; (3) o processo é demasiado rígido ou não está definido correctamente, i. e., os processos permanecem em "arquivo" ou então a abordagem é demasiado "alto-nível".

Sendo o metamodelo *Software Process Engineering Metamodel* (SPEM) [3] apresentado como norma para a especificação de processos, a sua comparação com os metamodelos dos processos existentes deverá evidenciar as suas analogias e divergências. Neste artigo investiga-se a expressividade e adequabilidade da terminologia SPEM para a especificação de processos. A comparação com os metamodelos de três processos - *Rational Unified Process* (RUP), *Extreme Programming* (XP) e *Microsoft Solutions Framework* (MSF) - permitirá determinar um conjunto de padrões observados durante o ciclo de vida dos processos. A selecção do RUP deve-se ao facto de ser um dos processos mais completo e divulgados. A escolha do XP relaciona-se com o facto de ser um dos processos ageís mais abordados nos ambientes estudados. Em relação ao MSF deve-se à importânica dada por este processo à equipa de trabalho. Não se enquadra nos objectivos do artigo uma descrição dos processos, para esse efeito deve-se consultar [4] [5] [6] [7] [8].

O facto dos padrões de desenho terem demonstrado o seu valor torna promissora a aplicação de técnicas semelhantes aos processos. Os padrões de processos [9] [10] fornecem orientações sobre a gestão de tarefas num processo, i. e., representam uma abordagem estruturada para tarefas que demonstraram resultados efectivos [11]. A definição dos padrões permitirá especificar uma arquitectura para os processos. O realizar desta análise tem por motivo o desenvolvimento futuro de uma ferramenta que permita configurar processos através da arquitectura desenvolvida de forma a permitir uma alteração temporal que inclua actualizações em actividades, produtos e papéis. Os modelos de processos actuais não se adequam à realidade empresarial, existem dificuldades de enquadramento com a estrutura das organizações. Outro objectivo será a definição de estrutura da organização em termos de modelos de competências individuais [12] [13] e colectivas [14] dos seus membros e posterior ligação aos modelos de processos. Os modelos de competências [12] descrevem características particulares, capacidades,conhecimentos e habilidades necessárias ao indivíduo ou grupo para realizar determinadas tarefas. Este artigo inclui apenas a análise dos metamodelos de processos, não abrange os modelos de competências.

A escolha do metamodelo SPEM como base para a nossa investigação deve-se essencialmente às suas seguintes características: (1) conceitos inerentes ao processo bem definidos em UML; (2) fácil compreensão e permitir a colaboração entre os intervenientes; (3) facilita a definição de condições e de dependências complexas; (4) capacidade para automatizar a execução de processos; (5) é extensível (e.g., pode ser adicionado um modelo de competências da organização); (6) permite a especificação de diferentes processos (e.g., XP [4] [5], RUP [6] [7], MSF [8], DMR [3][15], CMM [16] ou ISO [17]); (7) incorpora e reutiliza as boas práticas observadas em projectos que recorrem a processos.

O estudo realizado permite verificar que o SPEM incorpora conceitos necessários para a definição de padrões de processos. O objectivo de criar padrões deve-se essencialmente: (1) documentação sobre organização de processos; (2) metodologia agnóstica, pode ser aplicada independentemente da metodologia escolhida e (3) benefícios para a engenharia de processos.

Este artigo está estruturado em quatro secções. A Secção

2 descreve o metamodelo SPEM, focando sobre os conceitos, componentes e ciclo de vido do processo. A Secção 3 aborda alguns dos processos com maior aplicabilidade e apresenta, para cada um, a sua especificação baseada no metamodelo SPEM. A Secção 4 conclui o artigo descrevendo as características observadas nos vários processos que serão incorporadas nos padrões, acrescentando aspectos a considerar que não foram descritos.

2 METAMODELO SPEM

O SPEM [3] é um metamodelo normalizado com o objectivo de especificação dos processos. A actual versão do SPEM é a 1.0 definida com base no MOF 1.3 / UML 1.4 aguarda-se para breve a versão 2.0 baseada no UML 2.0. Sendo construído por extensão (SPEM_Extensions) de um subconjunto do metamodelo do UML 1.4, que se chama SPEM_Foundation [3], porém não se apresenta neste artigo o seu contéudo pois não serve de base para a comparação realizada.

A Fig1. mostra a estrutura interna do pacote SPEM_ Extensions, em termos de seus subpacotes e apresenta a sua dependência em relação aos subpacotes da SPEM_Foundation (Core, DataTypes, Model_Management, Actions, State_Machines e ActivityGraphs). Os seus constituintes são os subpacotes: BasicElements, Depedencies, ProcessStructure, ProcessComponents e ProcessLifecycle. Os quais adicionam os construtores e a semântica necessária à engenharia de processos.

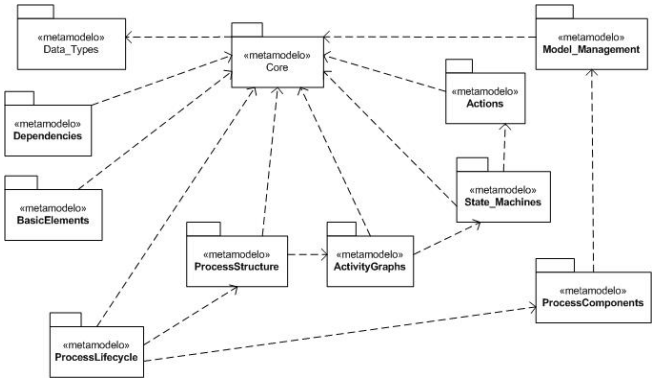

Fig. 1. Metamodelo das extensões para o SPEM.

Existem vários processos cuja definição já é baseada no SPEM, tais como, RUP (*Rational Unified Process*) [6] [7], DRM da Macroscope [15], *Global Services Method* da IBM [18] e *QuadCycle* da Unisys [19]. SPEM pode ser usado na descrição de padrões de processos tal como o UML é aplicado na descrição de padrões de desenho. Este metamodelo permite a definição formal de padrões organizacionais.

A Fig. 2 apresenta a arquitectura de modelação de quatro camadas proposta pela OMG [3]. A realização de um processo, i.e., a execução de um processo [20] associado a um projecto de desenvolvimento de software corresponde ao nível M0. A definição do processo associado, tal como o RUP, XP ou MSF, aparece no nível M1. O metamodelo, neste caso o SPEM, aparece no nível M2 e serve de "*template*" para o nível M1. A especificação SPEM é estruturada em UML e fornece um metamodelo completo baseado no MOF (*Meta-Object Facility*).

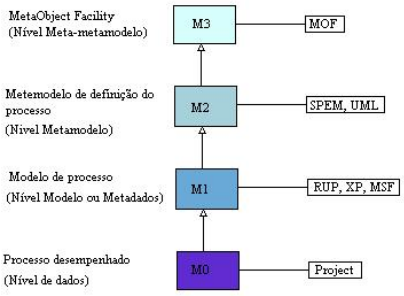

Fig. 2. Arquitectura de 4 camadas proposta pela OMG

Sendo o SPEM o metamodelo para especificação base de qualquer processo, como se demonstrará na secção 3, é necessário definir os seus constituintes básicos. Os princípios base do SPEM (Fig. 3) surgem da ideia que o processo é uma colaboração entre entidades abstractas activas designadas por papéis de processos (metaclasse process role) que realizam actividades (metaclasse activity) em entidades tangíveis e concretas chamadas produtos do trabalho (metaclasse work products).

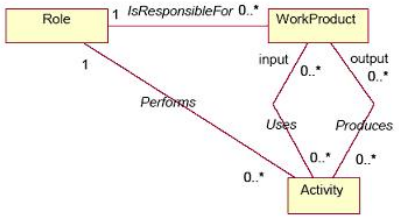

Fig. 3. Modelo conceptual do SPEM

Para adicionar informação detalhada ao processo, os elementos de **orientação** (metaclasse guidance do pacote BasicElements) são associados aos elementos do modelo. A classe tipo de orientação (metaclasse GuidanceKind) permite flexibilidade sobre os tipos de orientações usados num modelo de processo. Os **mentores de ferramentas** (metaclasse ToolMentor) são um tipo de orientação, demonstrando como realizar as actividades usando uma ferramenta específica. Cada mentor de ferramenta aparece associado a uma ferramenta específica e herda a associação à actividade que suporta através da classe orientação (metaclasse guidance).

2.1 Estrutura do processo

A Fig. 4 define os principais elementos de estrutura (subpacote ProcessStructure) que servem de base à construção de uma descrição de processo. Para uma integração nos conceitos apresentados na figura define-se previamente o conceito de core. O **core** que aparece em algumas classes referesse ao pacote que apresenta o conjunto de construtores essenciais ao metamodelo SPEM. O qual contem classes que não podem ser instanciadas (abstractas) mas definem conjuntos de características essenciais. Sendo o caso das classes modelelement, classifier e parameter. Estas classes abstrac-

tas servem de base a classes especializadas que podem ser instanciadas.

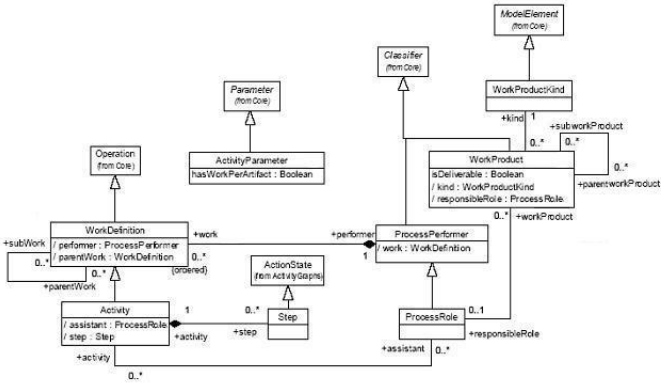

Fig. 4. Estrutura do processo (adaptado de [3])

No diagrama, a classe **produto de trabalho** (metaclasse work product) é algo que é produzido, consumido ou modificado através de um processo, pode ser um documento, um modelo ou um ficheiro com código fonte. O **tipo produto de trabalho** (metaclasse work product kind) descreve a categoria do produto, e. g., se é um documento de texto, um modelo UML, um ficheiro executável, uma biblioteca de classes ou outros. Pelo que, cada produto de trabalho tem associado um tipo produto do trabalho. A cada produto de trabalho pode ser associado um **papel** (metaclasse process role), o qual será formalmente responsável pela sua produção. Sendo produto de trabalho uma especialização da classe classifier pode participar na definição de um processo em associações e conter definições aninhadas.

A **definição de trabalho** (metaclasse work definition) é um elemento do modelo de um processo que descreve a execução, as operações e as transformações realizadas nos produtos do trabalho pelos papéis. A sua subclasse de maior impacto é actividade, mas fase, iteração e ciclo de vida são também suas subclasses (Fig. 6). Não se trata de uma classe abstracta, podem ser criadas instâncias para representar peças de trabalho que posteriormente serão decompostas. A relação entre as classes definição de trabalho e produto de trabalho é realizada através da classe parâmetro de actividade (metaclasse activity parameter) que especifica se o produto de trabalho é uma entrada ou um resultado da instância da classe definição de trabalho. Cada definição de trabalho tem um responsável que se designa por responsável de processo (metaclasse process performer).

A classe **responsável de processo** (metaclasse process performer) define o responsável por um conjunto de definições de trabalho de alto nível num processo onde não podem ser associados a papéis individuais. A sua subclasse **papel** (metaclasse process role) define responsabilidades e competências sobre produtos de trabalho específicos e indica os papéis que realizam ou assistem determinadas actividades. Sendo uma especialização da classe classifier pode participar na definição de um processo em relações de herança e associações.

A **actividade** (metaclasse activity) representa o trabalho realizado por um papel, correspondendo às tarefas, operações e acções realizadas ou coordenadas por um papel. Os elementos atómicos que constituem uma actividade designam-se por passos (metaclasse steps). Como um passo é classe derivada de estado de accção (metaclasse ActionState), o fluxo de passos numa actividade pode ser representado através de diagramas de actividades.

2.2 componentes do processo

A Fig. 5 descreve o pacote dos componentes (subpacote ProcessComponents) de um processo. As suas classes são responsáveis por dividir uma ou mais descrições de processos em partes "*self-contained*" que podem ser colocadas sob gestão de configuração e controlo de versões. Tal como no UML [21], um pacote pode conter os seus próprios elementos de definição do processo ou importar de outros. Este elemento e a dependência de categorização podem implementar o conceito de categoria para os elementos descritores do processo. Será criado um pacote para representar cada categoria e todos os elementos ligados por dependências de categorização serão membros dessa categoria. Um tipo de categorização de actividades é implementado em

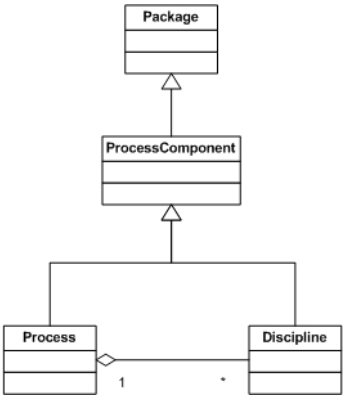

Fig. 5. Componentes do Processo

disciplinas.

A classe **componente de processo** (metaclasse process component) é uma descrição de processo que é internamente consistente e pode ser reutilizado por outros componentes do processo para criar um processo mais completo. Esta classe importa um conjunto de elementos de definição do processo (subpacote ModelElements). Neste conjunto não existem referências a depedências de membros do componente para elementos de outros componentes. O seu conteudo deve ser "internamente consistente" no sentido que as multiplicidades e restrições definidas no metamodelo devem ser satisfeitas no contexto desse componente.

O **processo** (metaclasse process) é um componente de processo que se comporta como um processo completo a partir do qual se podem especificar projectos. Distingue-se dos componentes de processo normais pelo facto que não é composto por outros componentes de processo.

Uma **disciplina** (metaclasse discipline) é uma especialização particular de pacote (metaclasse package) que agrega

as actividades de um processo segundo um "tema" comum. Esta organização das actividades implica que as orientações associadas e os produtos de trabalho de saída sejam categorizados segundo esse tema. A inclusão de uma actividade numa disciplina é representada pela dependência de categorização, com a restrição adicional de que qualquer actividade é categorizada por apenas uma disciplina.

2.3 ciclo de vida do processo

O pacote ciclo de vida (metaclasse Lifecycle) de um processo introduz os elementos de definição do processo que ajudam a representar a sua execução ao longo do tempo (Fig. 6). Estes elementos descrevem ou restringem o comportamento global de processo em execução e são utilizados para assistir no planeamento, execução e monitorização do processo. Um processo pode ser interpretado como uma colaboração entre papéis de modo a atingir um objectivo específico. Para coordenar a sua execução pode-se restringir a ordem de execução das actividades. Existe a necessidade de definir a "forma" do processo ao longo do tempo a sua

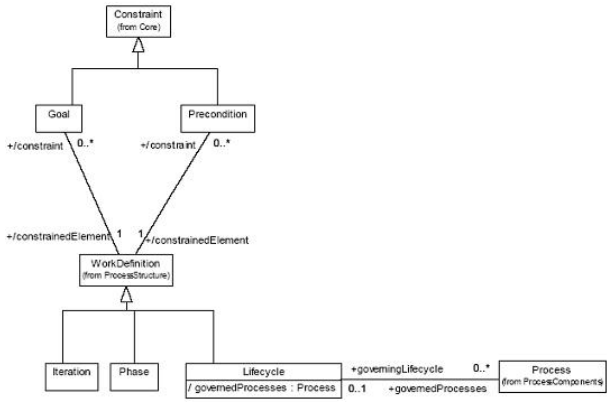

Fig. 6. Ciclo de Vida de Processo (adaptado de [3])

estrutura em termos de fases e de iterações.

A **fase** (metaclasse phase**)** é uma especialização de uma definição de trabalho (metaclasse work definition) de tal modo que a sua pré-condição (metaclasse precondition) define o critério de início da fase e o seu objectivo ou marco (metaclasse goal) define o critério de saída. Uma **iteração** (metaclasse iteration) é uma definição de trabalho composta mas com objectivos restritos (menores).

A classe **ciclo de vida** (metaclasse Lifecycle) encontra-se associada a uma sequência de fases com um objectivo específico. Este conceito define o comportamento de um processo completo a ser executado num determinado projecto ou programa. Na Fig. 7 pudemos observar a relação estrutural da classe ciclo de vida com as classes fase, iteracção, actividade e passo.

Cada definição de trabalho pode ter associada uma pré-condição e um objectivo, os quais são restrições escritas sobre a forma de expressões booleanas segundo uma sintaxe semelhante a uma condição-de-guarda do UML. A condição é expressa em termos do estado dos produtos de trabalho, os quais são parâmetro de definição de trabalho.

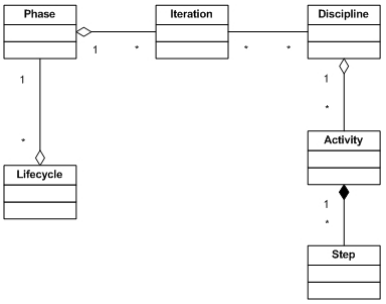

Fig. 7. Organização estrutural da classe ciclo de vida

3 PROCESSOS DE DESENVOLVIMENTO DE SOFTWARE

3.1 Rational Unified Process (RUP)

O RUP é um processo cuja base é um metamodelo que permite definir a linguagem (baseada no SPEM) que descreve o processo. Nesta secção descreve-se sucintamente o metamodelo do RUP. O modelo de organização do RUP (Fig. 8) centra-se nos elementos essenciais, definidos como estruturais e de comportamento [7]. A agregação desses elementos permite criar um conjunto de pacotes componentes de processo (metaclasse Process Component), os quais definem o modelo do processo RUP.

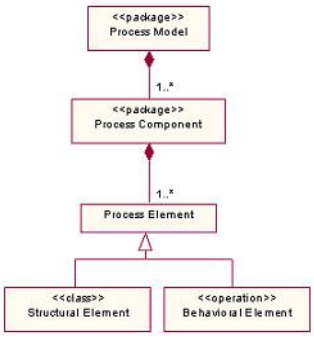

Fig. 8. Organização do RUP (adaptado de [7])

O modelo de descrição do metamodelo enumera os tipos de elementos do RUP (Fig. 9) e descreve as relações válidas entre esses elementos.

O **papel** (metaclasse role) corresponde ao papel de processo (metaclasse ProcessRole) do SPEM. No SPEM a noção de actividade tem exactamente a mesma designação e significado. O RUP inclui um conjunto extensão de papéis que são organizados nos seguintes grupos: analista, programador, gestor, testador, produtor e suporte, outros papéis. Os **artefactos** (metaclasse artifact), definidos no metamodelo SPEM por produtos de trabalho, são descritos no RUP segundo um conjunto de tipos de produtos de trabalho (metaclasse WorkProductKind), cada qual identificado por um estereótipo específico. Os tipos de artefactos válidos no RUP são: modelo (model), elemento de modelo (model element), documento (document), documento de especificação (specification document), repositório de dados (data store), documento de plano de trabalho (plan document), documento de avaliação (assessment document), executável (executable), infra-estrutura (infrastructure), genérico (ge-

neric).

No RUP existe um tipo de elementos, definidos por tipos de ficheiros de suporte, que permitem complementar os elementos principais com orientações adicionais sobre o processo. Contudo não constituem elementos do modelo do processo, não apresentando terminologia em UML. Através de uma ferramenta é possível criar e associar instâncias destes tipos de ficheiros a um ou mais elementos principais.

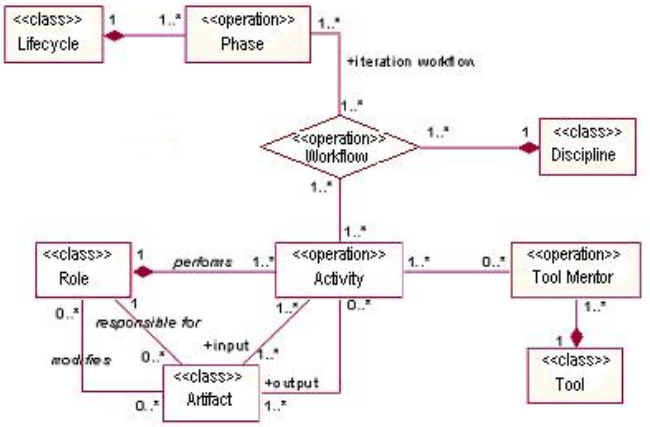

Fig. 9. Metamodelo do RUP (adaptado de [7])

A organização do processo permite aos interessados no projecto observar o RUP de diferentes perspectivas. Sendo construído um Website que permite navegar segundo um conjunto de vistas. De seguida descrevem-se alguns tipos de perspectivas típicas do processo RUP:

1. **Disciplinas**: cada disciplina (metaclasse discipline) apresenta um diagrama definindo o seu fluxo de trabalho (metaclasse workflow), o qual descreve como as actividades (metaclasse activity) são realizadas. As disciplinas do RUP são: Modelação do Negócio (Business Modeling), Requisitos (Requirements), Análise e Desenho (Analysis & Design), Implementação (Implementation), Teste (Test), Instalação (Deployment), Gestão de Configurações e Alterações (Configuration and Change Management), Gestão de Projecto (Project Management) e Gestão do Ambiente (Enviroment).
2. **Ciclo de Vida**: é importante quando se pretende planear actividades ou medir o progresso. Os elementos do ciclo de vida (metaclasse Lifecycle) definem a partição do tempo num conjunto de fases, sendo nomeadamente no RUP as **fases** (metaclasse phase**)** de: Incepção (Inception), Elaboração (Elaboration) Construção (Construction) e Transição (Transition).
3. **Papéis**: é útil para qualquer interessado, deve descrever os elementos do processo relevantes para determinado indivíduo. A vista de cada papel ilustra o conjunto de actividades realizadas ou os artefactos modificado por esse papel.
4. **Ferramentas**: Os mentores de ferramentas (ToolMentor) aparecem no RUP ligando as actividades a ferramentas (metaclasse Tool) como o Rational Rose, Rational RequisitePro, Rational ClearCase, Rational ClearQuest, Rational Suite TestStudio.

3.2 Extreme Programming (XP)

Os processos ágeis aplicam-se com especial relevância em pequenos projectos ou projectos com equipas de trabalho co-localizadas. Em comum com o RUP apresentam uma visão semelhante sobre as boas práticas necessárias ao desenvolvimento de software de qualidade, como por exemplo, o desenvolvimento iterativo e a preocupação nos requisitos e envolvimento dos utilizadores finais [4].

No processo XP (*Extreme Programming*) [4], a noção de **actividade** (metaclasse activity) (Fig.10) é mais próxima do contexto de disciplina (metaclasse discipline) do SPEM. As suas quatro actividades básicas são: planeamento (planning), desenho (designing), codificação (coding) e teste (testing). Estas actividades são realizadas segundo um conjunto de boas práticas que por vezes requerem a realização de actividades adicionais. O correspondente à actividade (metaclasse activity) do Metamodelo SPEM designa-se por **tarefa** (metaclasse task), sendo neste caso um elemento atómico que não é divisível. Os requisitos são apresentados pelos clientes em documentos designados por histórias (stories). Após a compreensão de uma história, os membros da equipa planificam as tarefas necessárias à implementação da história. As tarefas são realizadas por papéis (metaclasse role), os quais também são responsáveis por determinados artefactos, sendo designado por produto de trabalho na terminologia SPEM. Os **artefactos** (metaclasse artifact) são os produtos resultantes do processo, sendo as entradas e saídas das tarefas. A importância da documentação privilegiada pelo RUP deixa de ser um aspecto relevante no XP, já que este aspecto é preterido a favor da interacção entre os elementos da equipa e a sua estreita comunicação. Porém a quantidade de artefactos produzidos durante o processo é extensa, são alguns exemplos: as histórias (stories), restrições, testes de aceitação e unidades de testes, dados e resultados de testes, software, revisões (releases), desenho, normas de codificação, ambientes e ferramentas de teste, dados de métricas, relatórios e notas de reuniões.

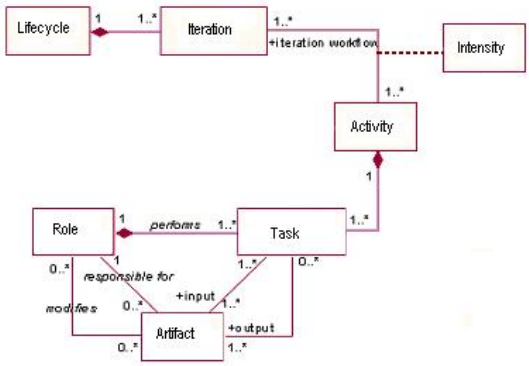

Fig. 10. Metamodelo do XP (adaptado de [7])

A um indivíduo ou grupo de pessoas podem estar atribuídos um ou vários roles. Não há correspondência unívoca entre papéis e intervenientes. Neste processo identifica-se sete **papéis** (metaclasse role): Programador (Programmer), Cliente (Customer), Testador (Tester), Rastreador

(Tracker), Treinador (Coach), Consultor (Consultant) e Chefe (Big Boss). Os papéis do XP são comparáveis a posições homólogas numa organização de desenvolvimento de software.

O desenvolvimento iterativo adiciona agilidade ao processo. A duração das **iterações** (metaclasse iteration), as quais definem o **ciclo de vida** (metaclasse lifecycle), permite avaliar o progresso de um projecto e realizar o seu planeamento de forma simples e viável. Em cada iteração, começa-se por definir o plano de acção com base nos artefactos planos de revisão (Release Plan) e erros (Bugs). Com base nestes elementos são definidas as tarefas de programação. Os elementos da equipa de desenvolvimento seleccionam as suas tarefas e estimam a sua duração. O número de iterações não é fixo, depende das histórias apresentadas pelos clientes e da planificação da equipa de desenvolvimento. No XP, o ciclo de vida de cada projecto inclui as iterações necessárias para que o trabalho seleccionado maximize o valor do negócio. As actividades (disciplinas no PSEM) repetem-se nas várias iterações, sendo definida a classe itensidade (metaclasse intensity) para as relacionar.

3.3 Microsoft Solution Framework (MSF)

O MSF (*Microsoft Solutions Framework*) [8] contem várias componentes que podem ser usadas individualmente ou de forma integrada: princípios de fundamento, modelos, disciplinas, conceitos chave, boas práticas e recomendações.

A falta de detalhes no MSF é uma característica que permitiu criar uma abordagem simples e directa. O Modelo da Equipa (metaclasse Team Model) e o Modelo do Processo (metaclasse Process Model) são descrições gráficas que mostram a lógica organizacional das equipas de projecto à volta de um grupo de papéis e as actividades ao longo do ciclo de vida do projecto (Fig. 11).

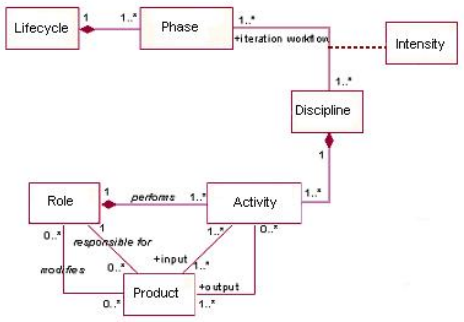

Fig. 11. Metamodelo do MSF (adaptado de [8])

O Modelo da Equipa (metaclasse Team Model) define os **papéis** (metaclasse role) e responsabilidades de uma equipa de pessoas trabalhando num projecto. Cada indivíduo da equipa pode ter atribuído um leque variado de papéis em áreas disciplinares interdependentes. Os seis papéis do MSF são: Gestor do Produto (Product Manager), Gestor do Programa (Program Manager), Programador (Developer), Testador (Tester), Gestor de Versões (Release Manager) e Avaliador da experiência do utilizador (User Experience). Cada projecto tem um ciclo de vida (metaclasse lifecycle), um processo que inclui todos os passos que ocorrem até à sua conclusão e transição para um estado operacional. A função principal do ciclo de vida é estabelecer a ordem de execução das **actividades** (metaclasse activity).

O Modelo do Processo (metaclasse Process Model) combina os benefícios dos marcos com as entregas iterativas e incrementais. É um modelo baseado em fases e marcos. O modelo de processos prevê 5 **fases** (metaclasse phase): Visão (Enviosioning), Planeamento (Planning), Implementação (Developing), Estabilização (Stabillizing) e Instalação (Deploying). Cada fase descreve um conjunto **produtos** (metaclasse products) que devem ser entregues, assim como marcos que devem ser atingidos e os respectivos critérios de aceitação [8]. Cada fase é vista como um período de tempo com ênfase em determinado tipo de actividades.

As **disciplinas** (metaclasse discipline) do MSF são áreas de prática que aplicam um conjunto de métodos, termos e abordagens específicas. As três disciplinas são: Gestão de Projecto (Project Management), Gestão de Risco (Risk Management) e Gestão de Competências (Readiness Management). As disciplinas perpetuam-se nas fases, sendo definida a classe itensidade (metaclasse intensity) para as relacionar.

4 ANÁLISE E DISCUSSÃO

A análise sucinta e comparação dos conceitos de base a cada um dos processos analisados permitem detectar a existência de elementos comuns, os quais apresentam correspondência no metamodelo SPEM. Numa perspectiva de evolução dos processos, os três processos analisados apresentam uma abordagem iterativa e incremental. Sendo uma característica dos processos mais recentes. Para uma comparação, observemos duas metodologias estruturadas com origem anterior aos modelos analisados: o Modelo em Cascata [1] e o Structured Systems Analysis and Design Methodology (SSADM) [22] [23].

Actualmente, a maior parte dos modelos de processos evolui a partir de três abordagens principais: Desenvolvimento Ad-hoc, Modelo em Cascata e Processo Iterativo. O Modelo em Cascata foi dos primeiros métodos estruturados, embora hoje seja muito criticado por ser muito rígido. Este modelo é constituído pelas seguintes disciplinas: Análise de Requisitos, Concepção (Análise e Desenho), Codificação, Testes e Manutenção. As actividades a executar são agrupadas em tarefas, executadas sequencialmente, de forma que uma tarefa se inicia quando a anterior terminar. O SSADM é um processo usado nas disciplinas de análise e desenho do desenvolvimento de software. Porém não integra modelação de processo, desenvolvimento, testes e implementação. A Tabela 1 permite avaliar a completude em termos de disciplinas dos três modelos iterativos analisados e dos Modelo em Cascata e SSADM.

TABELA 1
COMPARAÇÃO DAS DISCIPLINAS DOS CINCO PROCESSOS

	Modelação do Negócio	Requisitos	Análise	Desenho	Implementação	Teste	Instalação	Gestão de Configurações e Alterações	Gestão de Projecto	Gestão do Ambiente
RUP	√	√	√	√	√	√	√	√	√	√
XP	√			√	√	√			√	
MSF		√		√	√	√	√	√	√	√
Cascata		√	√	√	√	√	√			
SSADM		√	√	√						

Sendo o RUP um dos processos mais completos é natural que inclui uma maior variedade de disciplinas. O SSADM foi um modelo criado para integrar apenas as disciplinas de Análise e Desenho, sendo natural observá-lo integrado com outro processo. O XP é um processo ágil, como tal a interacção com o cliente é a chave para a validação dos requisitos. A sua maior ênfase é na programação. O XP é sobre desenho cuidado e contínuo, feedback rápido a partir de testes extensivos, manutenção clara dos aspectos relevantes e código de elevada qualidade. Sendo um processo ágil, o MSF especifica no Modelo de Equipa as preocupações relacionadas com o funcionamento da equipa de desenvolvimento.

Em síntese, todos os processos analisados apresentam disciplinas em comum, porém alguns são mais completos pois as exigências dos projectos a que se destinam requerem outro tipo de actividades, tais como: gestão de versões, gestão de qualidade e gestão de risco.

A Tabela 2 descreve a relação entre os conceitos dos metamodelos dos processos MSF, XP e RUP e os conceitos correspondentes no metamodelo SPEM.

TABELA 2
ELEMENTOS DE ESPECIFICAÇÃO COMUNS AOS TRÊS PROCESSOS E AO SPEM

SPEM	Ciclo de Vida	Fase	Iteração	Disciplina	Actividade	Passo	Papel de Processo	Produto de trabalho
MSF	Processo	Fase		Disciplina	Actividade		Papel	Produto
XP	Processo	Iteração		Actividade	Tarefa		Papel	Artefacto
RUP	Processo	Fase	Iteração	Disciplina	Actividade	Passo	Papel	Artefacto

As correspondências permitem definir padrões que representam a estrutura comum aos processos analisados. A documentação dos padrões [9] [10] é uma forma de partilha e reutilização da informação adquirida sobre os métodos apropriados para a organização e resolução de problemas ao longo do desenvolvimento de um projecto.

Os padrões [9] [10] deverão auxiliar durante todo o processo e segundo as perspectivas dos vários intervenientes. As vistas de um processo devem ser orientadas essencialmente por: ciclos de vida, disciplinas e papéis. Nesse contexto será necessário criar documentação sobre as disciplinas do projecto, i. e., identificar e especificar o seu conteúdo segundo um documento normalizado.

No que se refere ao ciclo de vida de um processo observámos algumas diferenças nos três processos analisados. A estrutura mais completa aparece no RUP que inclui três níveis: ciclo de vida, fase e iteração. O que se justifica pela sua aplicabilidade em projectos de maior complexidade. Os processos XP e MSF apresentam apenas dois níveis. Assim a definição do ciclo de vida deverá permitir uma estrutura dinâmica e configurável para o máximo de três níveis.

Os conceitos base dos três processos, actividades, papéis e produtos de trabalho devem apresentar um formato específico para a sua classe, sendo única para todos os processos. O ponto-chave será a interacção destes três elementos com as disciplinas e ciclo de vida. Estes elementos aparecem ligadas a disciplinas e partes do ciclo de vida específicas. A conexão deverá ser obrigatória, sendo no entanto definidos tipos de actividades, papéis e produtos de trabalho para restringir a liberdade de inclusão em grupos menos correctos.

Como conclusão podemos constatar que a definição de padrões de processos usando a terminologia do metamodelo SPEM é uma realidade que permite especificar vários tipos de processo. Inclusivamente, novos processos poderão resultar de alterações aos processos existentes. Existem vários tipos de processos, os quais não resolvem muitos dos problemas que se continuam a observar no desenvolvimento de software. Apesar do SPEM concordar com o metamodelo dos três processos analisados, não permite abordar os problemas observados actualmente. A solução passa por uma integração com a estrutura organizacional, onde se destacam os modelos de competências individuais e colectivas. Os padrões deverão ser definidos ao nível dos elementos do processo e da organização.

REFERÊNCIAS

[1] Roger S. Pressman, "Software Engineering – A Practitioner´s Approach", McGraw Hill, 5th Edition, December 2003

[2] Ian Sommerville, "Software Engineering ", Addison Wesley , 7th Edition , May 2004

[3] OMG, "Software Process Engineering Metamodel Specification", Version 1.0, Novembro 2002, http://www.omg.org/technology/documents/formal/spem.htm (acedido em Junho de 2004)

[4] K. Beck, "Extreme Programming Explained". Boston, MA: Addison-Wesley, 2000

[5] R. Jeffries, A. Anderson , C. Hendrickson, "Extreme Programming Installed", Addison Wesley , 1st Edition , October 2000

[6] P. Krutchen, "The Rational Unified Process: An Introduction", Addison-Wesley Pub Co, 3rd edition, December 2003

[7] Rational Unified Process (RUP), Rational Unified Process, Version 2003.06.01

[8] G. Lory, "Microsoft Solutions Framework", Version 3.0, http://www.microsoft.com/technet/itsolutions/techguide/msf/msfovrvw.mspx (acedido em Junho de 2004), 2003

[9] S. W. Ambler, "Process Patterns: Building Large-Scale Systems Using Object Technology", New York: SIGS Books/Cambridge University Press, 1998

[10] S. Ambler, "More Process Patterns: Delivering Large-Scale Systems Using Object Technology", New York: SIGS Books/Cambridge University Press, 1998

[11] J. Coplien, "A Generative Development-Process Pattern Language", Pattern Languages of Program Design, Addison Wesley Longman, Inc., pp.

183-237, 1995
[12] Sheryl Moinat, "The Basics of Competency Modeling", Version 1.2, Abril 2003
[13] T. Davenport, "Knowledge Management Case Study: Knowledge Management at Microsoft", Abril 1997
[14] Penna, "Team Competency Research Report", Novembro 2002
[15] DMR Consulting, "DMR Macroscope", Version 3.1, April 2000
[16] CMM, Capability Maturity Model for Software, http://www.sei.cmu.edu/cmm (acedido em Junho de 2004)
[17] ISO, ISO/ICE 12207, http://www.software.org/quagmire/descriptions/iso-iec12207.asp (acedido em Junho de 2004)
[18] IBM, Global Service Method, http://www-1.ibm.com/services/us/its/pdf/active_directory-011604.pdf (acedido em Junho de 2004)
[19] Unisys QuadCycle, "A full life cycle component based development and deployment methodology based on RUP and Unisys TeamMethod", http://www.unisys.com/news/releases/1999/oct/10276812.html (acedido em Junho de 2004)
[20] P. Barthelmess,"Colaboration and coordination in process-centered software development enviroments: a review of the literature", Information and Software Technology, Elsevier, 2003 pp. 911-928
[21] T. Pender, "UML Bible", Wiley Publishing Inc., 2003
[22] P. Weaver, et al, "Practical SSADM 4: A Complete Tutorial Guide", Prentice Hall, 3rd Edition, January 2002
[23] S. Skidmore,"SSADM Version 4 Models and Methods ", The Stationery Office Books, June 1997

Paula Ventura Martins Mestrado em Engenharia Informática em 2000 pela FCT/UNL, Licenciatura em Engenharia Informática em 1993 pela FCT/UNL. Analista de Sistemas no Instituto de Soldadura e Qualidade. Assistente no Departamento de Engenharia Electrónica e Informática da Universidade do Algarve, investigador convidado no INESC-ID na área de Sistemas de Informação. Interesse na área de processos de desenvolvimento de software, workflows, linguagens de modelação.

Alberto Rodrigues Silva Doutoramento em Engenharia Informática e Computadores pelo IST/UTL, mestrado em Engenharia Informática e Computadores pelo IST/UTL e licenciatura em Engenharia pela FCT/UNL. Professor auxiliar no Departamento de Engenharia Informática do IST/UTL, investigador sénior no INESC-ID na área de Sistemas de Informação e consultor informático em diferentes empresas e instituições.