EPIMA: uma abordagem para desenvolvimento de sistemas acadêmicos

Ana Paula Chaves[1], Gislaine Camila L. Leal[2], Jocimara S. Ferranti[3], Rogério S. Pozza[4]

[1] Coordenação de Informática – Universidade Tecnológica Feral do Paraná
Campus Campo Mourão – PR – Brasil
chavesana@gmail.com

[2] Departamento de Engenharia de Produção – Universidade Estadual de Maringá
Maringá – PR – Brasil.
gclleal2@uem.br

[3] Instituto de Informática e Estatística – Universidade Federal de Santa Catarina
Florianópolis – SC – Brasil
joccyferranti@gmail.com

[4] Coordenação de Informática – Universidade Tecnológica Federal do Paraná
Campus Cornélio Procópio – PR – Brasil.
pozza@utfpr.edu.br

Resumo Este artigo apresenta uma abordagem denominada EPIMA, baseado nas fases fundamentais do processo de software definidas na literatura. Esta abordagem foca em projetos acadêmicos de desenvolvimento de software, realizados em disciplinas de Estágio Curricular Supervisionado, que envolvem a construção de pequenos sistemas comerciais, com poucas funcionalidades. Dessa forma, a EPIMA contém um conjunto reduzido de fases que propõe auxiliar o desenvolvimento, desde sua especificação até sua conclusão, garantindo também sua manutenção. A abordagem foi aplicada no desenvolvimento de dois sistemas que, atualmente, se encontram implantados, caracterizando a resolução de problemas em contextos reais de aplicação.

Abstract This paper presents an approach called EPIMA, based on fundamental phases of software processes defined on the literature. The approach focuses on academic projects of software development, involving small commercial software building, with few features. In this sense, EPIMA contains a small set of phases aiming to support the development, since its specification until the develivery, assuring its maintenance. As a result of using this approach, EPIMA was applied on the development of two software that, nowadays, are being utilized, showing its applicability on real problem resolution.

1 Introdução

A crescente demanda por produtos e serviços de melhor qualidade tem impulsionado os mercados à obtenção de vantagens competitivas. Com o desenvolvimento de software não é diferente. O usuário, cada vez mais exigente, requer

sistemas que atendam suas necessidades e que sejam desenvolvidos no menor prazo possível. À medida que os sistemas de software crescem em complexidade, o processo de desenvolvimento também se torna mais complexo.

Vários modelos de processos definidos na literatura são propostos para atender a diversos tipos de sistemas com as mais variadas necessidades. Esses modelos sugerem a realização de uma determinada seqüência de atividades para que o desenvolvimento do software seja bem sucedido.

Entretanto, na formação de profissionais, existe uma dificuldade em ensinar os alunos sobre a importância da aplicação desses processos. Isso porque os software desenvolvidos em disciplinas curriculares são geralmente pequenos, formados por poucas rotinas ou funcionalidades. Por esta razão, os alunos preferem simplesmente codificar, ao invés de "perder tempo" realizando documentação. Essa visão é compartilhada por alguns profissionais de pequenas empresas, que desenvolvem sistemas sem seguir um processo adequado, o que posteriormente pode provocar dificuldades de manutenção e melhorias, como pode ser visto em [1].

Este artigo tem como objetivo propor uma abordagem simplificada, baseada em características de modelos já existentes, porém, possuindo um conjunto reduzido de fases, com atividades bem definidas e gerando uma quantidade restrita de artefatos. O objetivo da abordagem é mostrar aos alunos das primeiras disciplinas de programação e análise, a importância de ser guiado por um processo durante o desenvolvimento de software, mesmo que para software de pequeno porte.

Este artigo está organizado em 5 seções, além desta: a Seção 2 expõe a metodologia utilizada para o desenvolvimento desta pesquisa; a Seção 3 apresenta os conceitos de processos de software e suas fases essenciais; a Seção 4 descreve a abordagem EPIMA, objetivo principal deste artigo; a Seção 5 apresenta e discute a aplicação da abordagem EPIMA em dois projetos de desenvolvimento de sistemas em disciplinas de Estágio Curricular; e, por fim, a Seção 6 traz as conclusões e trabalhos futuros.

2 Metodologia

A pesquisa conduzida neste trabalho se caracteriza como sendo de base qualitativa e do tipo exploratória. A metodologia utilizada no desenvolvimento deste trabalho é composta por três fases: definição do problema, especificação da solução e prova de conceito.

Na fase de definição do problema, foram analisadas as dificuldades apresentadas por alunos da disciplina obrigatória de Estágio Curricular Supervisionado do curso de Tecnologia em Desenvolvimento de Sistemas de Informação, da Universidade Tecnológica Federal do Paraná, Campus Cornélio Procópio. Essa disciplina envolve, em grande parte, o projeto e desenvolvimento de pequenos software comerciais. Cada aluno propõe o desenvolvimento de um software, estando vinculado a uma determinada empresa que funciona como cliente. O aluno

desenvolve o sistema sob a orientação de um professor e a supervisão de um funcionário da empresa, cumprindo um cronograma predeterminado de 400 horas.

Um dos objetivos do estágio é proporcionar a adaptação do estudante à sua futura atividade profissional. Portanto, torna-se fundamental que compreendam a importância da utilização de processos de desenvolvimento para guiar suas atividades, aplicando técnicas de engenharia de software que propiciem o aumento da qualidade dos sistemas desenvolvidos. Entretanto, observou-se que os sistemas desenvolvidos na disciplina de estágio são, em geral, pequenos em termos de quantidade de rotinas ou funcionalidades. Por esta razão, embora os alunos desse período curricular já conheçam as técnicas de Engenharia de Software, o desenvolvimento do sistema é geralmente caótico, baseado apenas em requisitos-programação. Os alunos não seguem qualquer abordagem que guie o desenvolvimento do sistema, já que a grande parte dos processos encontrados na literatura são focados em projetos mais complexos, envolvendo diversas fases e grande quantidade de artefatos que, para sistemas acadêmicos, nem sempre são necessários.

Assim, o resultado dessa fase da pesquisa identificou a necessidade de uma abordagem simplificada, que produza apenas os artefatos fundamentais para guiar o desenvolvimento de um sistema com poucas funcionalidades, e que permita aos alunos de cursos de desenvolvimento de sistemas a compreender a importância da utilização de processos de software.

A fase de especificação da solução corresponde à elaboração da abordagem EPIMA, apresentada neste artigo e, mais detalhadamente, descrita na Seção 4. A fase de prova de conceito consistiu na aplicação da abordagem EPIMA em dois projetos de disciplina de Estágio Curricular Supervisionado. O resultado desta fase está descrito na Seção 5.

3 Processo de Software

Um processo de software pode ser definido como um conjunto coerente de políticas, estruturas organizacionais, tecnologias, procedimentos e artefatos necessários para conceber, desenvolver, implantar e manter um produto de software [2]. Envolve uma série de etapas constituídas por um conjunto de atividades, métodos, práticas e tecnologias que são utilizadas desde o desenvolvimento até a manutenção do software e produtos relacionados.

Um processo pode ser imaturo e utilizar de uma abordagem ad-hoc para organizar o trabalho, ou maduro, sendo caracterizado por uma metodologia padronizada e documentada, que já tenha sido utilizada em projetos similares ou, que seja, habitualmente, adotada em uma organização. Segundo [3], o uso de um processo padrão permite aos gerentes de projeto definir planos em conformidade com os padrões de qualidade e procedimentos da organização.

Uma equipe de desenvolvimento que trabalha sem um processo bem definido acaba funcionando de maneira ad-hoc, o que compromete o seu sucesso, criando uma situação insustentável. Por outro lado, organizações maduras empregando

um processo bem definido podem desenvolver sistemas complexos de maneira consistente e previsível, independente de quem o produziu [4].

Além disso, o processo deve possibilitar a melhoria da qualidade de serviço, de engenharia e de projeto; a diminuição de custos por meio do aumento da previsibilidade e capacidade de mitigar riscos, bem como a melhoria da eficiência e produtividade da organização.

Uma arquitetura de alto nível para o ciclo de vida de software é proposta pela norma ISO/IEC 12207 pela definição de processos, atividades e tarefas que podem ser aplicadas na aquisição, no fornecimento, no desenvolvimento, na operação e na manutenção de software. Cada processo é composto por atividades, e cada uma destas possui um grupo de tarefas associadas.

A ISO/IEC 12207 define 17 processos, os quais estão classificados em: processos fundamentais, processos de apoio e processos organizacionais. Os processos fundamentais compreendem os processos envolvidos na execução do desenvolvimento, operação e manutenção do software durante o ciclo de vida, além da contratação entre cliente e fornecedor. Os processos de apoio colaboram com o sucesso e a qualidade do projeto de software. Os processos organizacionais são empregados por uma organização para estabelecer e implementar uma estrutura constituída pelos processos do ciclo de vida e pelo pessoal envolvido no desenvolvimento de software.

O modelo de um processo de software é a representação formal dos elementos e características envolvidos neste processo. Segundo [5], um modelo de processo deve especificar os pré-requisitos e conseqüências de cada tarefa, bem como a sincronização entre as mesmas.

A modelagem de um processo de software deve ser conduzida de modo a possibilitar o entendimento e a padronização do processo. [6] ressaltam que a representação do processo possibilita a comunicação e o entendimento efetivo do processo, facilita o reuso, apoia a evolução, facilita o gerenciamento do processo e é importante na avaliação, evolução e melhoria contínua do processo.

A estrutura de um modelo de processo de software é, geralmente, composta por quatro elementos:

- Ator: representa as entidades que executam um processo ou assumem um papel durante a execução de uma tarefa;
- Papel: grupo de responsabilidades que executam uma atividade específica do processo de software;
- Artefato: porção representativa de informação do produto que resulta de uma atividade e pode ser utilizado posteriormente como matéria-prima para gerar novos artefatos;
- Atividade: consome artefatos (de entrada) e gera novos artefatos (de saída).

Os processos de software, em sua maioria, são bem estruturados e possuem fases bem definidas, que por sua vez, compreendem uma série de atividades que são executadas para que um objetivo seja atingido. Algumas dessas fases, entretanto, são comuns aos diversos processos existentes, por exemplo, a Especificação, o Projeto e a Implementação, e podem ser consideradas fundamentais para o desenvolvimento de um produto de software.

Diversas classificações foram propostas por vários autores. De acordo com [7], as quatro fases principais são:

- Especificação: definição das funcionalidades e restrições do software;
- Projeto e implementação: produção do software de acordo com a especificação;
- Validação: garantia de que o software satisfaz as necessidades do usuário;
- Evolução: adaptação do software de acordo com novas necessidades.

[8] simplifica o processo de software apresentando apenas três fases, porém, com características similares: Definição, Desenvolvimento e Manutenção.

Embora as fases sejam encaradas como passos discretos e seqüenciais elas não impõem seqüencialidade obrigatória em sua execução. As fases realizadas e a forma com que são executadas dependem da aplicação, ambiente e objetivo do processo.

4 EPIMA (Especificação, Projeto, Implementação e MAnutenção)

De acordo com a norma ISO/IEC 12207 a EPIMA corresponde a um processo fundamental de desenvolvimento, cujo objetivo é apoiar a transformação dos requisitos de um usuário em um produto de software.

EPIMA é uma proposta baseada nas fases fundamentais de um processo de software, apresentadas na Seção 3. Possui quatro fases, divididas em um conjunto restrito de atividades, que possibilitam a construção de um sistema desde sua especificação até sua conclusão, incluindo sua manutenção. Baseado no paradigma de orientação a objetos, utiliza a UML (*Unified Modeling Language*) na modelagem dos artefatos e permite a iteração das atividades com o objetivo de refiná-los. Alguns artefatos são sugeridos como opcionais, cabendo a quem instancia a abordagem decidir por sua construção ou não, de acordo com a necessidade do sistema.

Além destas características, é importante destacar que, como o processo foi modelado para apoiar o desenvolvimento de aplicações de nível acadêmico, pressupõe-se que todas as atividades serão realizadas pelo mesmo indivíduo. Portanto, na abordagem EPIMA, todos os papéis são desempenhados por um único ator, não havendo distinção entre atores desenvolvedores, analistas, gerentes ou qualquer outra função.

As seções 4.1 a 4.4 apresentam a descrição detalhada de cada fase e as atividades compreendidas.

4.1 Especificação

A fase Especificação consiste na definição do escopo do sistema, propondo soluções para o problema identificado, e no levantamento dos requisitos necessários. Executa as seguintes atividades:

1. Estudo do Negócio: nesta atividade são realizadas as entrevistas com os usuários e os estudos referentes à área de aplicação do sistema que será desenvolvido;
2. Definição do Escopo: o objetivo desta atividade é definir os limites e as restrições do sistema e identificar os atores que interagem com ele. Os artefatos gerados são a Descrição Formal do Sistema, que consiste em uma descrição textual das principais funcionalidades e a Definição dos Atores, que detalha o papel que cada ator do sistema, seja ele usuário ou aplicação externa, representa para o sistema.
3. Detalhamento dos Requisitos: de acordo com o resultado da atividade anterior é realizado um refinamento das funcionalidades, possibilitando o detalhamento dos requisitos funcionais que o sistema deve compreender. Os artefatos produzidos são a Especificação dos Casos de Uso, que descreve em linhas gerais as funções dos casos de uso, e o Diagrama de Casos de Uso, que representa o sistema por meio da perspectiva do usuário. A construção de Diagramas de Atividades, que descreve os passos a serem percorridos para a conclusão de um caso de uso, é opcional, já que essa descrição pode ser feita textualmente na Especificação dos Casos de Uso.

4.2 Projeto

Na fase de projeto é realizada a definição conceitual dos elementos necessários para que o sistema seja implementado. Seu objetivo é documentar os objetos pertencentes à aplicação e seu comportamento. Suas atividades são detalhadas abaixo:

1. Definição dos Objetos: com base na definição dos casos de uso, esta atividade define os objetos necessários ao sistema e a relação existente entre eles. Produz como artefato o Diagrama de Classes, que define a estrutura das classes utilizadas no sistema e serve de apoio para a construção da maioria dos outros diagramas.
2. Definição do Comportamento: esta atividade modela os aspectos comportamentais do sistema e a forma como reage aos eventos externos. Os artefatos são Diagramas de Sequência, que representa a ordem temporal em que as mensagens são trocadas entre os objetos, e Diagramas de Colaboração, que modelam a forma como os objetos estão vinculados e quais mensagens trocam entre si.
3. Modelagem das Entidades: entidades são os objetos persistentes do sistema. Esta atividade tem como finalidade modelar esses objetos e os relacionamentos existentes entre eles. Como artefato, o Modelo Entidade-Relacionamento (MER), que descreve o modelo de dados de um sistema com alto nível de abstração, é mapeado de acordo com as classes identificadas como persistentes no Diagrama de Classes documentado.

4.3 Implementação

Nesta fase é implementado todo o código que compõe o sistema e, à medida que cada funcionalidade é construída, são realizados os testes para garantir o atendimento dos requisitos e a coerência dos resultados gerados. Após a codificação das funcionalidades as partes são integradas. Suas atividades são descritas a seguir:

1. Codificação: implementação dos casos de uso baseada no comportamento descrito pelos Diagramas de Seqüência. O artefato desta atividade é o código-fonte das classes que compõem o sistema.
2. Testes: paralela à codificação, a atividade de teste consiste em avaliar cada funcionalidade implementada. O método utilizado é o teste de caixa preta, realizado para demonstrar que cada função do software é totalmente operacional [8]. Alguns protótipos podem ser gerados como artefatos desta atividade a fim de auxiliar na validação dos resultados.
3. Integração: junção das funcionalidades desenvolvidas, gerando um sistema homogêneo. Esta atividade resulta no produto de software concluído, adequado para ser implantado no ambiente do usuário.

4.4 Manutenção

Esta fase compõe um ciclo iterativo que abrange todas as fases anteriores com objetivo de adequar novas funcionalidades e/ou corrigir falhas identificadas após a implantação do sistema no ambiente do usuário. Propõe a construção do Manual do Usuário, que busca auxiliar o usuário a identificar cada uma das funções do sistema, e do Manual de Instalação, que tem como objetivo conduzir o usuário durante a instalação, evitando que este dispenda mais esforços do que os necessários. Estes artefatos são opcionais, já que alguns sistemas possuem funcionalidades indutivas e/ou não requerem quaisquer detalhes de instalação.

A Figura 1 representa graficamente a EPIMA. O plano Fases, como o próprio nome sugere, demonstra as fases da abordagem, enquanto Especificação, Projeto, Implementação e Manutenção descrevem suas atividades, conforme definidas nas Seções 4.1 a 4.4.

As setas indicam o fluxo de execução. As atividades do plano Especificação são seqüenciais. Em Projeto, a Modelagem das Entidades e a Definição do Comportamento ocorrem depois da Definição dos Objetos, independentemente da ordem de execução. As atividades Codificação e Testes, em Implementação, podem ser executadas paralelamente, seguidas pela Integração. E por fim, em Manutenção, um ciclo iterativo abrange as fases anteriores, de acordo com as necessidades identificadas.

O usuário do sistema participa de todo o processo, porém, com maior presença nas fases de Especificação e Implementação. Na primeira, o usuário tem papel fundamental durante o Estudo do Negócio, caracterizando a situação problema, e a Definição do Escopo, avaliando a solução proposta. Na segunda, auxilia na atividade de Testes, e ao fim da Integração, verifica a conformidade do sistema implementado com os requisitos definidos inicialmente e a consistência dos resultados apresentados.

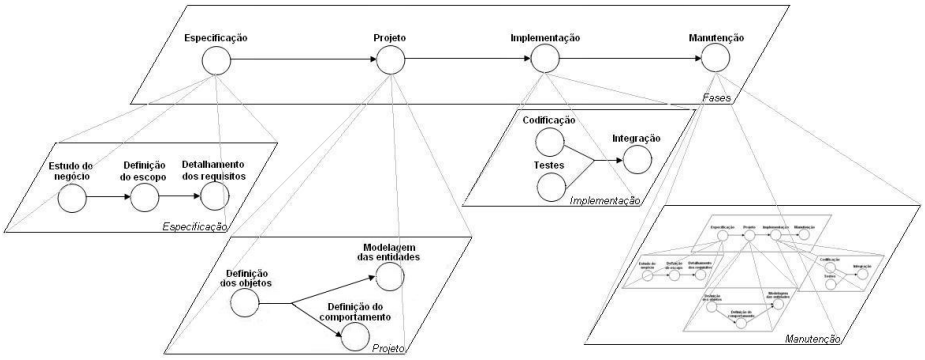

Figura 1. EPIMA

5 Prova de Conceito

Este Seção apresenta dois exemplos de aplicação da abordagem EPIMA e serve como prova de conceito. Esse tipo de estudo é realizado por meio da descrição do funcionamento da abordagem em um determinado exemplo, que pode ser real ou não.

A prova de conceito caracteriza-se pela falta de formalização durante a realização do estudo, o autor não utiliza os conceitos de experimentação, o que inviabiliza obter conclusões sobre a real viabilidade e funcionalidade da abordagem [9].

A EPIMA foi aplicada a um cenário real que consistia na construção de dois sistemas por duas alunas do curso de Tecnologia em Desenvolvimento de Sistemas de Informação, do Campus de Cornélio Procópio da Universidade Tecnológica Federal do Paraná (UTFPR), na disciplina de Estágio Curricular Supervisionado, em parceria com a unidade de Londrina da Empresa Brasileira de Pesquisa Agropecuária (Embrapa) – Embrapa Soja.

Os sistemas foram desenvolvidos pelas alunas individualmente, fora da sala de aula, e as restrições, sugeridas pelo corpo técnico da empresa parceira, onde os sistemas estão implantados. O desenvolvimento foi realizado com uma carga horária de aproximadamente 400 horas. As seções 5.1 e 5.2 apresentam a descrição dos sistemas, com seus objetivos, tecnologias e ferramentas utilizadas. A Seção 5.3 discute alguns fatores importantes acerca da aplicação da abordagem.

5.1 Ocorrências do Sistema de Alerta

O Ocorrências do Sistema de Alerta é um módulo de um sistema WEB responsável por manipular informações referentes à dissipação da ferrugem asiática pelas várias regiões produtoras do Brasil. Além disso, o módulo disponibiliza, com o apoio de uma ferramenta *OpenSource*, MapServer, o mapa do país destacando os municípios registrados como foco da doença.

As informações manipuladas, como o município onde o foco foi identificado, o tipo de área da plantação, a fase de desenvolvimento da planta, o laboratório responsável pela confirmação da ocorrência e as condições climáticas da região afetada, entre outras, mostram aos técnicos e produtores as características da plantação contaminada.

As Figuras 2, 3 e 4 apresentam, respectivamente, o Diagrama de Casos de Uso, o Diagrama de Classes e o Modelo Entidade-Relacionamento do módulo Ocorrências do Sistema de Alerta.

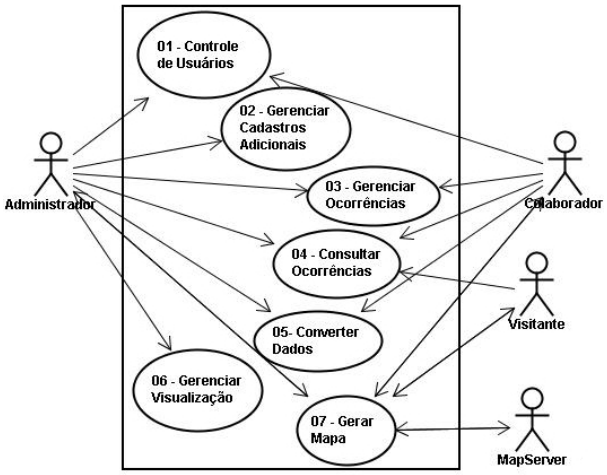

Figura 2. Diagrama de Casos de Uso - Sistema de Alerta

5.2 Desdobra Segregantes

O Desdobra Segregantes é um sistema *stand-alone* para apoiar o processo de melhoramento genético de sementes de trigo na Embrapa Soja. O objetivo é automatizar a fase de geração de históricos de sementes, armazenar as informações geradas e permitir que esses dados sejam manipulados. Além disso, o sistema possibilita a importação/exportação de informações de/para arquivos Excel (.xls) do pacote Microsoft Office.

O software foi construído baseado no paradigma de orientação a objetos, com linguagem de programação Java. Para a persistência dos dados, utilizou-se o banco de dados MySQL, na versão 4.1. As ferramentas que apoiaram o desenvolvimento foram a IDE Netbeans 4.1 para a implementação, Jude Community 1.5.2 para modelagem de processo e ERwin 4.1 para modelagem de dados.

As Figuras 5 e 6 apresentam, respectivamente, o Diagrama de Sequência e o Modelo Entidade-Relacionamento do sistema Desdobra Segregantes.

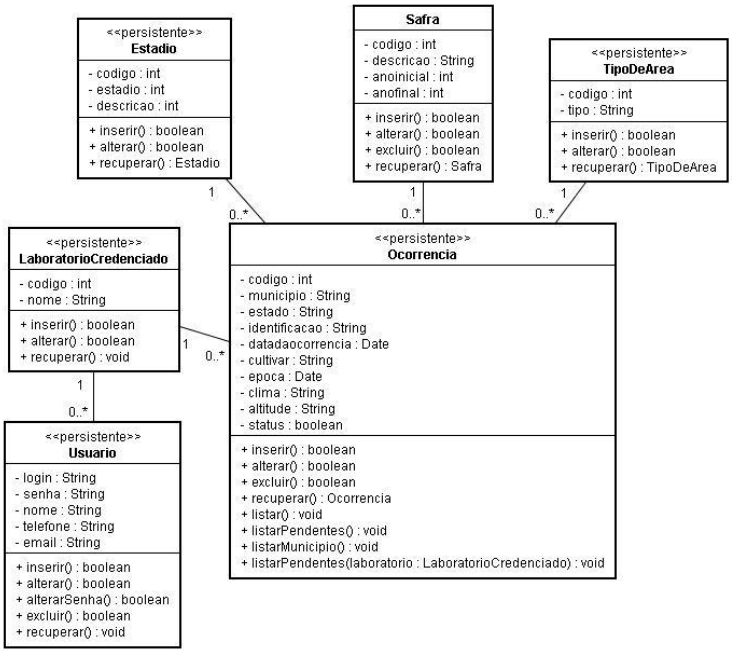

Figura 3. Diagrama de Classes - Sistema de Alerta

5.3 Discussão

Em ambos os sistemas em que a EPIMA foi aplicado, a fase que exigiu maior esforço foi a Especificação. Para que os Casos de Uso pudessem ser especificados passo a passo, foi necessário um grande entendimento acerca do negócio de aplicação. Foram realizadas várias entrevistas na empresa para analisar as rotinas e necessidades dos usuários, já que esse refinamento de requisitos é a base para todas as demais fases do desenvolvimento. Dessa forma, ressalta-se que a possibilidade de iteração das atividades de cada fase permite que as informações e artefatos sejam refinados de acordo com a necessidade da aplicação.

O fato do desenvolvedor, durante a aplicação da abordagem, optar por não construir os artefatos opcionais não resulta no mau êxito do sistema. O módulo Ocorrências do Sistema de Alerta teve seus Diagramas de Atividades documentados, enquanto o sistema Desdobra Segregantes optou por não construí-los. Além disso, o Desdobra Segregantes apontou a necessidade da criação do Manual de Usuário e do Manual de Instalação. No entanto, as aplicações resultantes apresentam o mesmo nível de satisfação e completeza, de acordo com os usuários finais.

Finalmente, a aplicação da EPIMA em dois sistemas que seguem abordagens distintas, um stand-alone e um módulo WEB, enfatiza a idéia de reutilização da abordagem em processos de desenvolvimento que possuam características partic-

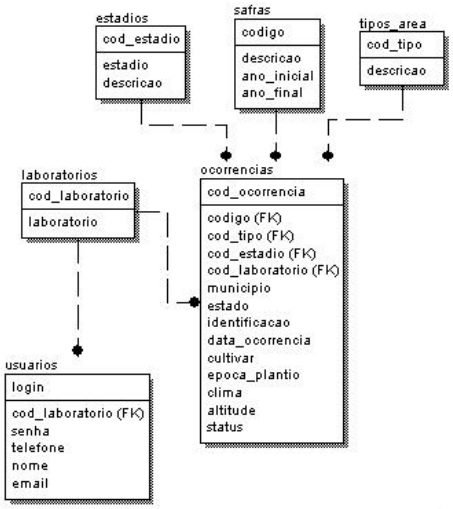

Figura 4. Diagrama de Entidade-Relacionamento - Sistema de Alerta

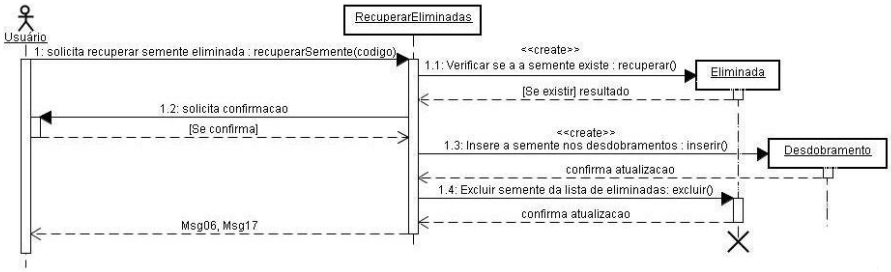

Figura 5. Diagrama de Sequência - Desdobra Segregantes

ulares e se utilizem de tecnologias diferentes, como linguagens de programação e interfaces de comunicação.

6 Considerações Finais

O presente artigo teve por objetivo apresentar a abordagem EPIMA, baseada nas fases fundamentais do processo de software. Sua proposta é proporcionar que os alunos da disciplina de Estágio Curricular Supervisionado sigam alguma sequência de atividades durante o desenvolvimento de seus sistemas, mostrando a eles os benefícios de se seguir uma abordagem e iniciando-os na aplicação de técnicas de instanciação de processos e modelos de qualidade.

Após a definição da metodologia e a descrição dos conceitos de processo de software, foi exposta a abordagem EPIMA, com detalhamento de suas fases,

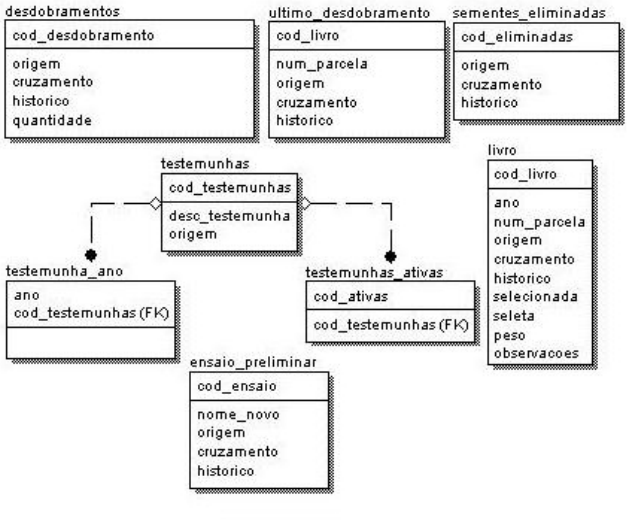

Figura 6. Diagrama de Entidade-Relacionamento - Desdobra Segregantes

atividades e artefatos. Por último, são descritos dois sistemas onde a abordagem foi aplicada.

A contribuição deste artigo é a proposta de uma abordagem simplificada, com um conjunto reduzido de fases, atividades e artefatos, mas que é suficiente para o desenvolvimento de sistemas comerciais pequenos, com baixa complexidade de implementação. A abordagem auxilia os alunos a, mesmo durante o desenvolvimento de seu primeiro projeto, compreender a importância da utilização de processos para o desenvolvimento e manutenção de sistemas de qualidade. Vale ressaltar que, nesta abordagem, todos os papéis são desempenhados por um único ator, não abrangendo processos executados por equipes de desenvolvimento estruturadas, sendo adequado para sistemas a serem desenvolvidos por um único indivíduo, desde a especificação até a conclusão, como é o caso dos sistemas acadêmicos.

A aplicação da EPIMA no desenvolvimento de dois pequenos sistemas, apresentada na Seção 5, possibilitou uma prova de conceito da abordagem e apontou algumas limitações passíveis de melhoria. Alguns pontos são ressaltados como possibilidades de extensão desta pesquisa:

- Realizar uma avaliação da abordagem, utilizando os conceitos da engenharia experimental, para verificar o impacto da utilização da abordagem sobre a aprendizagem do aluno;
- Incluir elementos dos processos de apoio, que possibilitem o gerenciamento do processo, auxiliando o controle de custos, recursos e o acompanhamento de cronogramas, visando a qualidade do software;

– Modelar a abordagem utilizando uma notação adequada, como o SPEM, o que possibilita sua evolução e manutenção e comunicação;
– Acompanhar o desenvolvimento de projetos acadêmicos que utilizam a EPIMA, com o objetivo de detectar as deficiências da abordagem, o que contribui para identificar possíveis pontos de melhoria.

Referências

1. von Wangenheim, C.G., Anacleto, A., Salviano, C.F.: Mares - a methodology for software process assessment in small software companies. Technical Report LPQS001.04E, Universidade do Vale do Itajaí – UNIVALI (2004)
2. Fuggetta, A.: Software process: a roadmap. In: ICSE '00: Proceedings of the Conference on The Future of Software Engineering, New York, NY, USA, ACM (2000) 25–34
3. Berger, P.M.: Instanciação de processos de software em ambientes configurados na estação taba. Master's thesis, COOPE, Universidade Federal do Rio de Janeiro, Rio de Janeiro - Rio de Janeiro (2003)
4. Kruchten, P.: The Rational Unified Process: An Introduction. Addison-Wesley (2003)
5. Ben-Shaul, I.Z., Kaiser, G.E.: A paradigm for decentralized process modeling and its realization in the oz environment. In: ICSE '94: Proceedings of the 16th international conference on Software engineering, Los Alamitos, CA, USA, IEEE Computer Society Press (1994) 179–188
6. Humphrey, W.S., Kellner, M.I.: Software process modeling: principles of entity process models. In: ICSE '89: Proceedings of the 11th international conference on Software engineering, New York, NY, USA, ACM (1989) 331–342
7. Sommerville, I.: Software Engineering. 5th edn. Addison-Wesley (1995)
8. Pressman, R.S.: Engenharia de Software. 5 edn. Mc Graw Hill (2002)
9. Kitchenham, B.A., Pfleeger, S.L., Pickard, L.M., Jones, P.W., Hoaglin, D.C., Emam, K.E., Rosenberg, J.: Preliminary guidelines for empirical research in software engineering. IEEE Trans. Softw. Eng. **28**(8) (2002) 721–734