Modelagem e Design do Projeto

Processo de projeto

O processo de projeto, seja de um sistema qualquer de Engenharia e especialmene de um sistema de informação consiste de algumas etapas genéricas - naturalmente estas etapas podem ser detalhadas à luz de diferentes métodos - mostradas na figura abaixo.

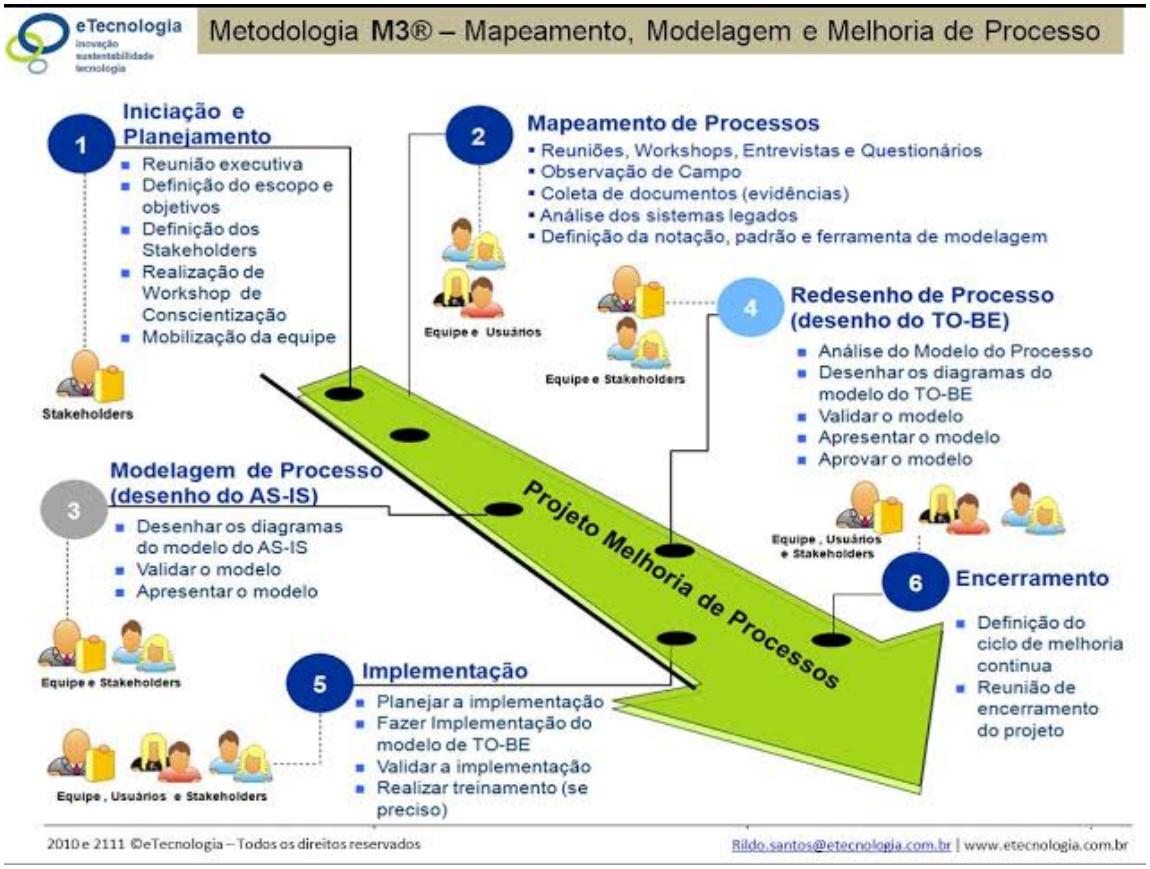

Figure 1: Processo de projeto usado em empresas de Sistemas de Informação, seja para criar um novo projeto ou prover a melhoria de processos já existentes.

Como estamos nos dedicando a um projeto simples, feito por uma equipe reduzida e com um cronograma de tempo relativamente curto, vamos seguir um modelo simplificado do processo de projeto. Mesmo assim, algumas etapas precisam ficar muito claras e fundamentar a documentação, elemento essencial em qualquer projeto.

Na Fig. 1 vemos um exemplo esquemático dos passos para um processo de projeto aplicável a sistemas de informação. Este modelo foi extraído de empresas embora seja plenamente válido academicamente, mas, certamente não teremos que seguir (e nem é possível) todos estes passos. O item 1 é pertinente ao planejamento do projeto e é o início de tudo. Da lista de itens teremos apenas a definição de escopo e objetivos - expresso no primeiro documento de projeto na forma de use-case - e, o que pode ser acrescentado neste modelo, a definição de stakeholders. Mesmo que não seja o caso de entrevistar ou "ser contratado por stakeholder", esta definição pode ser importante para definir classes de agentes e/ou interfaces (*views*). Portanto

vamos partir do pressuposto que os objetivos e o escopo estão bem definidos no documento já submetido no e-disciplinas.

No item 2 praticaremos somente a definição de notação (glossário de termos). Este glossário deve ser inserido na seção 2 do documento de projeto já existente (elaborado à partir do template). Note que o responsável pela documentação deve zelar pela consistência e atualidade deste glossário de termos e conceitos. Toda a equipe de projeto deve ser referir a esta parte do documento, o que assegura que todas "falam a mesma língua", sem problemas de desvio de interprestação.

A definição da ferramenta de modelagem é importante e vamos neste curso adotar o UML2 como base de representação. Portanto ferramentas como Modelio (www.mdelio.org), Astah (astah.net), Eddraw (www.eddrawsof.com), StarUML (staruml.io) ou UML Designer (www.umldesigner.org) podem servir de base no caso da nossa disciplina. Novamente estas são ferramentas consgradas no mercado: i) Modelio é uma ferramenta mais pesada e usada em projetos europeus; Astah é hoje uma ferramenta comercial e deve maner um trial; Eddraw é considerada a mais "user friendly" do mercado e também tem trial; StarUML é open source e funciona em macOS, Windows e Linux; UML Designer é uma ferramenta completa e trabalha com UML 2.5 e com linguagens de especficação. Todas funcionariam e até têm bem mais recursos do que teremos que usar nesta disciplina. Precisamos portanto definir um deles, já que uso da UML já está definida.

O próximo passo é definir quais diagramas usar (a UML 2.5 tem pelo menos 12 diagramas com sub-diagramas). Você pode ainda preferir ter uma visão sistêmica e diagramática dos casos de uso já documentados. Isso ajuda a ver com clareza todos os agentes e elementos que devem ser representados por classes. Ajuda também a ver os principais views the devem ser elaborados para estes diferentes agentes, com suporte no "model" que você deve preparar em Ruby. A ferramenta Visual Paradigm (www.visual-paradigm.com) tem download gratuito e é simples e direta. Ferramentas diferentes? na verdade o diagrama de casos de uso não pertence aos diagramas básicos da UML (embora tenha grande uso prático e apareçam em várias ferramentas). Este diagrama foi elaborado por Ivar Jacobson para capturar requisitos, não o design de sistemas. E esta é de fato a fase do projeto que você está na Aula 4 de laboratório. Estes diagramas de casos de uso também podem ser incorporados à documentação e são a expressão dos objetivos e escopo do projeto. Portanto terminamos a fase 2 (mapeamento de processos) enriquecendo a documentação e com uma visão mais ampla do sistema de informação a ser criado.

Uma referência interessante sobre modelagem de casos de uso está no site www.methodsandtools.com, onde existe um arquivo Understanding Use Case Modeling. Um livro clássico sobre o assunto é Use Case Driven Object Modeling with UML, de Doug Rosenberg e Kendall Scott, editado pela Addison-Wesley. Esse livro tem o aval da OMG (Object Management Group). Você pode acrescentar este diagrama na seção 2 do documento.

O item 3 trata da modelagem do processo (e todo sistema de informação o que faz é justamente automatizar, resgistrar, e controlar processos, geralmente discretos). Agora vamos de fato modelar os processos, o que implica também em prover as interfaces e views. Cada processo (e o seu sistema pode ter vários) é um elemento de design baseado em model-view-control. O fluxo básico (veja item 2.1 da documentação) pode ser representado por um diagrama de estados da UML, já que representa os estados do seu sistema e sua evolução. Em uma primeira abordagem o que importa é o relacionamento estado-transição. Para saber se uma dada transição está habilitada precisamos saber quais os pré-requisitos e, uma vez ocorra os pós-requisitos que passarão a ser verdadeiros. Exceções e casos especias podem dar margem a fluxos alternativos que devem ser igualmente incorporados à documentação. Vejam que agora estamos em um nível bem mais detalhado e mais próximo da implementação (embora ainda não possa relacionar nada com o código Ruby).

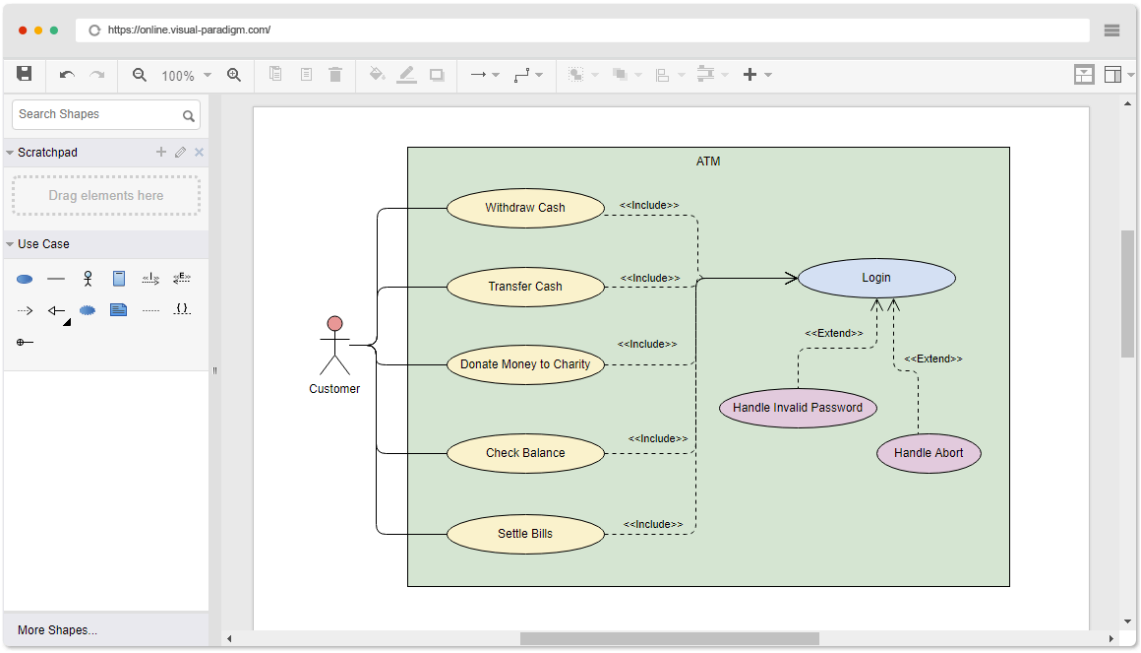

Figure 2: Captura de tela do Visual Paradigm para representação de diagramas de casos de uso.

Modele cada um dos processos, inclusive o processo básico. Você pode acrescentar digramas (de estado, de atividades) associado a cada um, é opcional, mas é imprescindível inserir as informações pedidas no template, e que vão também orientar e documetar o código na fase de implementação. A seção 3 do documento de projeto deve receber esta informação, com ou sem diagramas. Os diagramas facilitam a visualização, pense nisso.

Mas tem um detalhe! será importante inserir uma modelagem dos dados (de forma estática). As ferramentas sugeridas acima modelam diagramas entidade-relacionamento (diagramas ER). Os relacionamentos entre dados ficam mais claros também. Durante a iplementação esta será uma referência importante. Na verdade estamos ainda na fase de modelagem e este é de fato um modelo entidade-relacionamento (MER), sempre presente em qualquer projeto de sistemas e particularmente de sistema de informação.

O MER foi criado por Peter Chen em 1976, época áurea do desenvolvimento de ferramentas e métodos para lidar com bancos de dados relacionais. Simples e prático este modelo foi incorporado a quase todas as ferramentas de modelagem conceitual de bancos de dados (relacionais). Passou também a ser incorporado ao design de sistemas, indo além da modelagem de dados. Aqui a idéia é inspirar a criação do modelo de dados, que foi exercitado nas aulas anteriores, sem nenhum design ou model prévio (até porque não seria necessário nos exemplos vistos). Mas você pode agora visualizar o modelo dos exercícios usando um diagrama ER.

O modelo criado até aqui deve passar pela fase 4, isto é, deve ser revisado inteiramente antes da validação. Esta validação seria feita diretamente com um "stakeholder" e/ou com usuários típicos do sistema, o que não é o caso na disciplina PMR3304. Portanto vocês devem improvisar a validação com os prórios membros da equipe. Possíveis erros ou imprecisões, dados ou processos incompletos, ou sem um fluxo claro devem ser corrigidos, modificados ou eliminados. O seu projeto *muda de versão*. Veja no template da documentaçã que tem um cabeçalho que precede o conteúdo do documento onde se declara o nome do projeto, a versão corrente, a data em que foi homologada (importante para saber a versão corrente - acordem o sujeito que controla a documentação!). Uma vez satisfeitos com a versão corrente (a que deve

ser apresentada no final do curso) odem seguir adiante para a implementação do código em Ruby on Rails.

Um lembrete: use a seção 3.4.3 (requisitos especiais) para inserir os requisitos não funcionais, isto é, requisitos de performance, segurança e outros requisitos especiais pertinentes ao seu projeto.

Sobre a implementação vamos fazer uma pequena melhoria: até aqui programamos na linha de comando, de modo a ficar absolutamente acima de plataformas, versões de sistemas operacionais, etc. Foi um bom exercício conceitual, sem a interferência de especificades de ferramentas. Mas, para a implementação de um projeto, que mesmo sendo um exemplo de sistema de informação simples, é bem maior que os exercícios das aulas anteriores, precisaremos ter uma ferramenta com um suporte e visualização melhor da arquitetura RoR (Ruby on Rail). Sugerimos o uso de uma ferramenta que tem um bom suporte mas que, por outro lado não acrescenta muito na curva de aprendizado em termos de detalhes da própria ferramenta: o RubyMine.

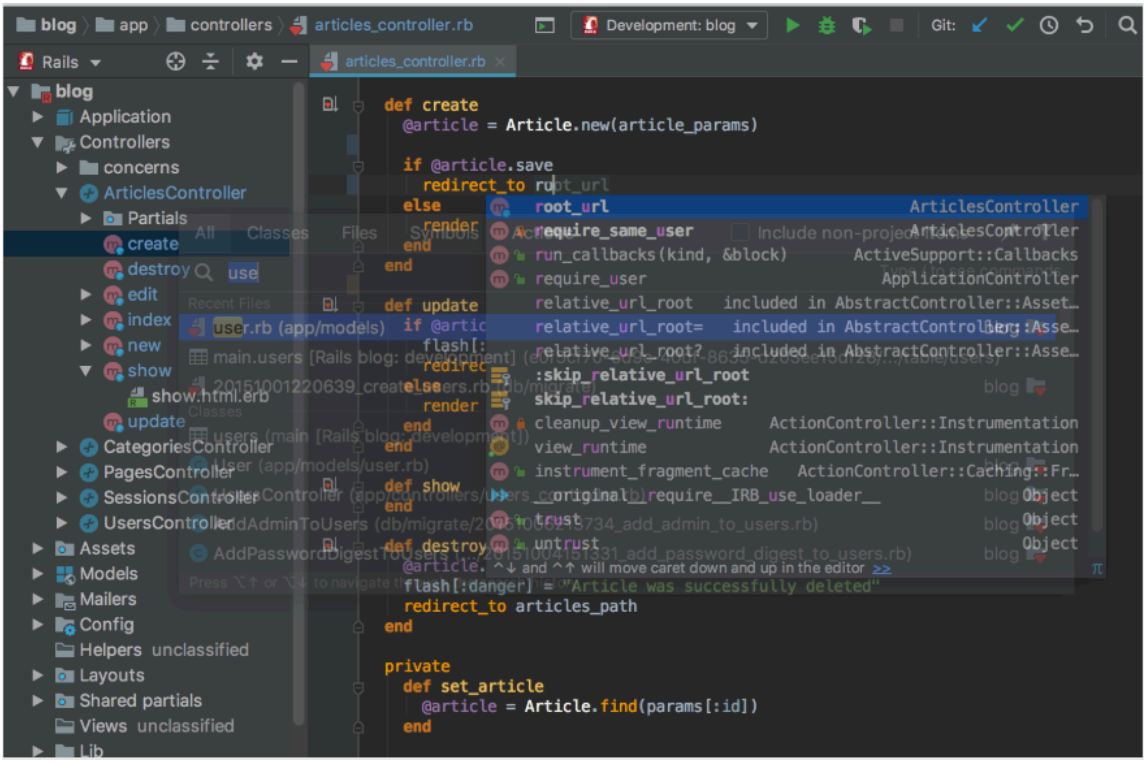

Figure 3: O RubyMine também é uma ferramenta de uso corrente em empresas e ambientes de desenvolvimento de código em RoR.

O RubyMine é uma ferramenta da Jetbrains, e tem download gratuito apenas para instituições de ensino (alunos e professores). Você deve preencher a uma requisição de licença, usar o seu e-mail com domínio "usp" e a licença será atividade com um e-mai que é enviado imediatamente após o pedido.

Instale o RubyMine e experimente agora desenvolver o código usando os conceitos das aulas anteriores mas a interface do RubyMine ao invés da linha de comando. Pra começar você poderá ver toda a arquitetura do Ruby on Rails em volta da tela de comando, o que torna as coisas mais simples. O RubyMine é apenas uma ferramnta do tipo IDE, você ter o Ruby já instalado na máquina e a versão correta do sql3. Em um documento à parte daremos mais suporte e

instruções de como vincular seu projeto e instalação do Ruby a um banco de dados Posgree.

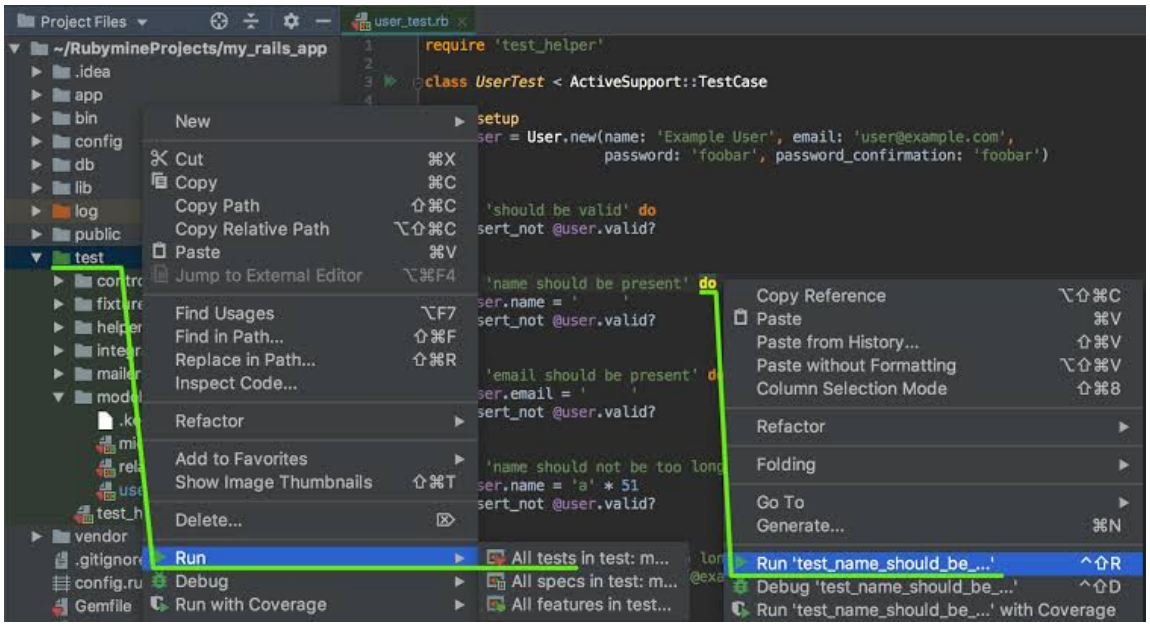

Figure 4: O RubyMine em pleno funcionamento executando um teste.

Os principais aspectos da implementação de um aplicativo simples de sistemas de informação foram dados nas aulas anteriores. Isto praticamente encerraria o projeto exceto por um detalhe: você deve ter feito o código e testado o fluxo dos processos em separado. Isso se chama teste local. Agora precisaria fazer uma nova batelada de testes do sistema como um todo: o teste global.

O resultado desse teste pode simplesmente atestar que o projeto está concluido e dá-se o encerramento mostrado na fase 6 da Fig. 1. Entretanto é provável (até bastante provável) que existam correções a serem feitas, adendos (se for verificado que os objetivos do projeto não foram plenamente atendidos), etc. Isso implica em retornar à fase 4 da Fig. 1. Claro, não é necessário fazer tudo novamente, mas preparar uma nova versão do sistema (o colega da documentação continua acordado, certo!). Quando este ciclo se encerra sem mais adendos (ou quando decidirem que chega!) o projeto está terminado (na atual versão, porque requisitos mudam com o tempo, especialmente em sistemas de informação). Vocês devem preparar a versão final para demonstração.

Finalmente...

Tendo agora um projeto (que imagino deve funcionar) é tempo de preparar um pequeno ”manual de usuário”. A função de ter este trecho no projeto seria ter documentado o funcionamento padrão de modo que possamos acompanhar o seu projeto sem ter que necessáriamente ler o código ou verificar o diagrama de estados (não que seja impossível fazer isso). Mas, pensando em termos mais práticos e na vida ”depois de formado” temos que dizer que FAZ PARTE da documentação de projeto um trecho que unifique a expecatativa de funcionamenteo geral do sistema. Entre outras utilidades isto serve quando se faz o teste global mencionado acima, especialmene se este envolve usuários ou pessoas que não pertemcem à equipe de desenvolvimento.

Assim, a interação entre sistema e usuários ou stakeholders deve ser prevista, implementada e plenamente documentada. Use a seção 4 do template de documentação para isso. Ainda que as ações não se restrinjam a inserir dados de produto, como feito no exemplo de implementação e no template padrão da documentação, TODAS as ações que estejam previstas nos objetivos do projeto (e este é um momento para verificar isso mesmo sendo uma abordagem informal) devem estar presentes nesta parte da documentação indicando que estes objetivos foram de fato atingidos, verificados e testados.