Features

Aqui voce encontra

Todas informacoes sobre o conceito de *features* utilizado no Processo 3PUP. Para conhecer os tipos agregadores, consulte Tipo Theme, Tipo Set e Tipo Epic.

Conteúdo:

Arquivos de configuracao

Para quem usa o Enterprise Architect, baixe aqui o arquivo de configuracao da ferramenta.

Para usa-lo:

1. Abra o EA
2. Crie um novo modelo em branco
3. Importe o arquivo, via menu Tools - Import Reference Data
4. Crie suas features 😉

1. Definição

A definição de feature no projeto 3PUP é a seguinte:

> *Uma feature é um elemento de projeto que representa o menor valor agregado para o projeto ou negócio do cliente.*

Observa-se aqui três coisas importantes:

- Primeiro, a palava *menor* que denota a indivisibilidade da feature, ou seja, ela não pode ser decomposta em frações menores sem perder o conceito de se apresentar como algo útil que agregue valor para alguém.
- Segundo, justamente o conceito de *valor agregado* que é o porquê de alguém querer ela. Em outras palavras, até poderiam existir coisas que são pequenas e indivisíveis, mas se não agregarem valor, elas não são features.
- Terceiro, perceba as palavras *projeto* ou *negócio*. Isso significa que as features podem existir tando para atender necessidades específicas e solicitadas pelo cliente (como um botão de tela) mas também para atender demandas do projeto por si só, como as que ocorrem atrás da grande cortina (uma refatoração de código-fonte para melhorar a legibilidade do programador, por exemplo). Este é terceiro ponto é um dos motivos que as features são categorizadas em tipos, a fim de se medir, acompanhar e melhor gerenciar o projeto, permitindo um senso de balanceamento entre esforço empreendido para entregar features que agreguem de fato valor ao cliente, daquelas que existem para manter o projeto em andamento (o famoso overhead operacional ou peso do processo).

Logo, a feature é como um átomo, que se for fracionada, perde a propriedade essencial que a definie, que é o *valor agregado*, seja ao projeto ou ao negócio.

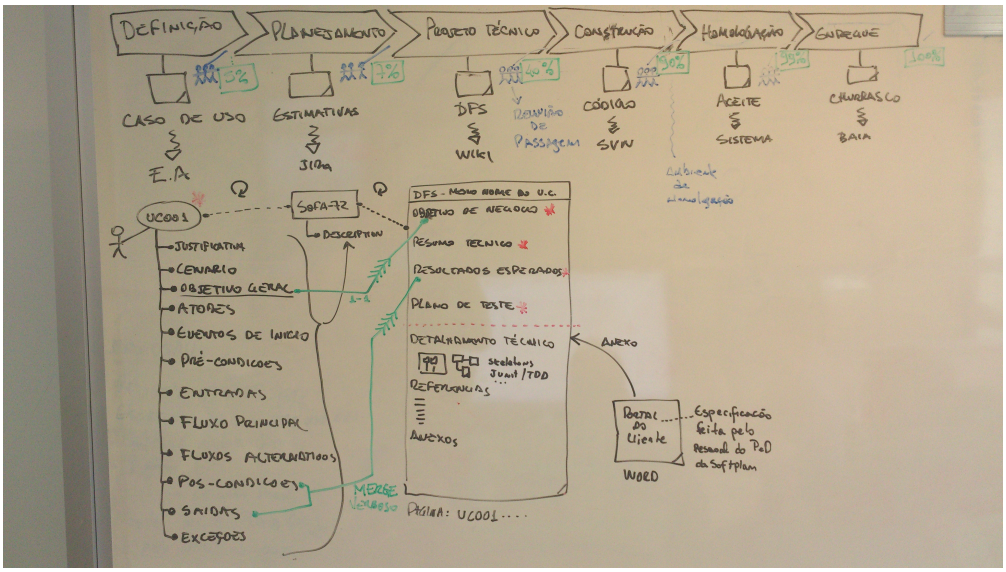
Portanto, features precisam ser pequenas, singelas, decididamente claras e finitas. Qualquer feature que não atenda esses requisitos não é, de fato, uma feature, e precisa ser revisada.

Os próximos capítulos exploram mais as características das features.

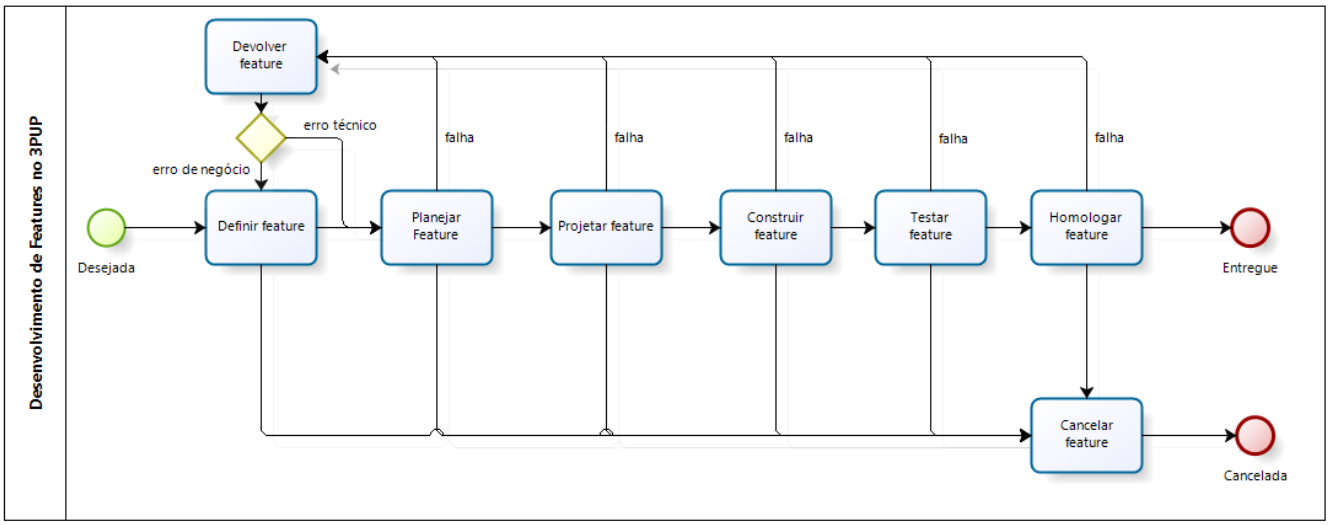
2. Ciclo de Vida

Abaixo, o ciclo de vida resumido e detalhado de uma feature normal (clique para expandir):

2.1.1. Idéia Original

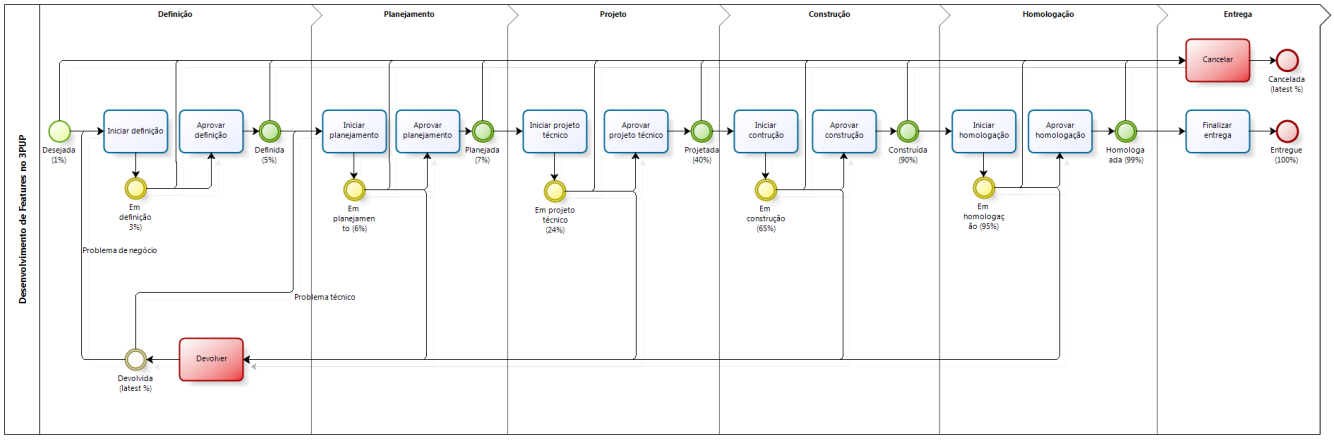


2.1.2. Visão Conceitual



2.1.3. Visão Detalhada

⚠️ AVISO: A imagem abaixo deve ser ajustada, para incluir a atividade de teste após a construção.



Os status das features sao assim descritos:

O Time e o Product Owner trabalham da definição da feature, buscando saber em que parte do projeto este "algo" se encaixa, ou mesmo se ela vai ser uma Feature mesmo ou então um módulo novo, um Serviço, um Set ou um fragmento de funcionalidade que precisa ser incorporado em uma Feature já existente ou, em outra vertente, se este algo não vai fazer parte do projeto ou mesmo se ele vai ser um projeto em separado!

Para sair desse status, a Feature precisa:

a) Ter um nome (no padrao A.A.R.O.)
b) Ter um local para ser encaixada (Modulo, Servico e Set)
c) Ter uma descricao geral, em um parágrafo.
d) Ter um dono (isso deriva do padrao AARO acima, e indica que ela tem um usuario ou grupo de usuarios que vao utiliza-la no sistema quando o projeto for entregue - ou seja, uma role que identifca quem utliiza a feature)
e) Ter um conjunto de requisitos que a balizam (podem ser textuais, genericos, mesmo sem grande formalismo, mas que norteiem claramente sua complexidade e escopo - lembre-se, quanto mais tempo for dedicado para isso nessa fase, mais subsidios ter-se-ao na etapa seguinte, de planejamento).

Nesse momento o PO, o Time e o ScrumMaster vao detalha-la, e ela somente pode sair desse status se:

a) Tiver um dono no projeto (um Chief Programmer da FDD - geralmente um analista de negocio responsavel ou um projetista responsavel), que vai ser responsavel por ela, respondendo por ela, durante todo o ciclo de vida do projeto
b) Ter um tamanho, mensurado em Feature Points (o que sugere que também é mapeado seu tamanho em horas-homem)
c) Ter uma data de entrega (o que sugere que vai ter um FixVersion de entrega). Nada impede que esse FixVersion seja "Unscheduled", ou seja, nao se sabe quando ela vai ser entregue - mas como se diz no PMBOK - "sabemos que nao sabemos quando ela vai ser entregue".
d) Ter uma prioridade (ou Rank no Scrum) perante as outras features no projeto, ou seja, ela tem um indice de importancia e interesse para o PO.

Nesta etapa de projeto, a feature será projetada, arquitetada e esmiuçada nos mínimos detalhes, pois é com base nesse projeto técnio que ela será construída. Nesse ponto, a feature precisa estar exatamente como o PO a deseja.

O projeto técnico envolve tanto o Time (na figura, geralmente, do analista ou projetista responsável - o Chief Programmer, mas também envolvendo os outros recursos do time, como arquiteto, scrum master, desenvolvedores, e outros), quanto - e principalmente - o PO, e portanto ambos trabalham em sitonia, buscando detalhar cada aspecto relevante para a Feature representar o fidedigno do que o PO deseja.

O projeto técnico está completo, e a Feature somente pode sair desse status se:

a) Possuir um Caso de Uso, descrito segundo o padrão OBJETIVO; ATOR PRINCIPAL, ENVOLVIDOS E EXPECTATIVAS; PRE-CONDICOES; EVENTOS DE DISPARO; FLUXO PRINCIPAL, FLUXOS ALTERNATIVOS; POS-CONDICOES; EXCECOES)
b) Possuir uma lista de requisitos associados
c) Possuir um projeto de dominio, que descreve as Classes e Operacoes relacionadas (geralmente, um Diagrama de Classes)
d) Possuir um projeto de interface de usuario (geralmente uma tela, mas nem sempre), que identifique claramente os pontos de interacao e elementos envolvidos e tambem a parte navegacional do sistema (a IU em relacao as outras IU do sistema, menus, telas, validacoes, etc).
e) Possuir um projeto de implantacao e comunicacao, que descreve como a Feature deve ser empacotada e disponitilizada no sistema frente a estrutura e arquitetura da aplicacao.
f) Possuir um projeto de teste associado, incluindo teste unitario (a Feature funcionando), teste integrado (a Feature funcionando em conjunto com outras Features dentro do seu Servico), e funcional (a Feature em uso dentro do sistema final, sendo utlilizada pelo usuario final do sistema)

Observa-se que nesse ponto, o projeto tecnico segue o padrao 4+1 (porem ajustado), ou seja, a parte 1 é o quadrante logico (o Caso de Uso, os Requisitos e os Testes), a parte 2 é o nível de desenvolvimento (o Projeto de IU e o Diagrama de Classes), a parte 3 é o quadrante físico (o Diagrama de Implantacao ou Comunicacao) e a parte 4 é o quadrante de processo (ou de processamento, que envolve os famosos requisitos nao funcionais e a questao arquitetural, onde na verdade, o projeto tecnico da Feature precisa ser encaixado na arquitetura da aplicacao da forma mais transparente possivel, e por isso esse quadrante pode ser minimizado no projeto tecnico - por isso diz-se 4+1 ajustado, de forma que ele pode ser interpretado e implementado como uma lista de requisitos diferenciado - esteriotipo "nao funcional" por exemplo - na propria lista de requisitos da Feature.).

Caso o projeto técnico mostre discrepância em relação ao planejado (complexidade, tempo, etc), á Feature precisa retroceder no worfklow. Isso pode ocorrer na primeira tentativa de projeto técnico, ou mesmo depois que este já foi finalizado uma vez e a construção tenha sido iniciada ou mesmo já em validação (etapa de validação). Logo, fica subentendido que uma Feature nunca vai estar fora de sintonia em relação ao plajemaneto do projeto, seja estimada "para mais" ou estimada "para menos".

É responsabilidade do Chief Programmer "sentir" essa discrepância e reportar ao ScrumMaster e PO essa divergência para que a Feature volte para a etapa de planejamento. Somente com esse projeto tecnico, esta apta a passagem para a próxima etapa de construção.

Aqui, o Time trabalha continuamente, produzindo a feature, testando-a continuamente e evoluindo sua funcionalidade constantemente.

A presenca do PO aqui também é válida e muito importante, pois ele vai ajustando arestas e complmenetando informações durante a construção da Feature, o que vai minimizar quaisquer problemas na etapa posterior de validação funcional.

O Chief Programmer (ou Projetista ou Analista responsável) é o recurso que constanmente monitora e "puxa" o desenvolviemento da feature junto ao time, reportando o andamento diariamente durante as reuniões Stand Up de avaliação do Scrum.

Durante a construção é plausível que o projeto técnico mostre-se inadequado. Então, nesse caso, descarta-se totalmente o avanço na construção e esta deve retornar para o projeto técnico. Observa-se assim que a Feature pode retroceder no worfklow. Assim, não deve ser construída uma Feature que esteja fora de conformidade com o seu projeto técnico. Logo, Features que não podem ser construídas de forma avessa ao seu projeto técnico (seja porque o projeto técnico é inconsistente ou incompleto, seja porque ela está agredindo outra feature, seja porque o PO não está satisfeito com a Feature ou qualquer outro evento). Em todos esses casos, a Feature retrocede para o projeto técnico. É responsabilidade do Chief Programmer em conjunto com o Time reportar essas divergências de implementação frente ao projeto técnico.

A feature é dada como construída está apta para evoluir no workflow de validacao se, e somente se:

a) Todos os testes unitários estiverem consistentes.
b) Todos os testes integrados estiverem consistentes.
c) O PO deu seu OK na Feature durante o processo de homologação do sistema.

Nessa etapa, o PO em conjunto com seus usuários finais realiza os testes de aceitação da Feature, os chamados testes funcionais.
Nesse ponto, é esperado que estes testes sejam balizados pelo Casos de Uso associado à Feature, bem como a lista de requisitos que a norteiam, o projeto de IU associado e os testes funcionais atrelados.
Caso qualquer inconsistência apareça, a Feature precisa retroceder no workflow. É responsabilidade do PO, em cojunto com os usuários finais e o Chief Programmer realizar estes testes.

Qualquer anomalia ou inconsistência de funcionamento no teste funcional deve produzir um item de correrção de Feature, e ser ligado (linkado) para entrar no pipeline de correção do projeto.

Uma Feature somente pode sair da etapa de validação se:

a) Obter um aceite formal do Product Owner

Apos o periodo de uma entrega (1 sprint) no status anterior se nenhum problema ocorrer nesse meio tempo, a feature pode ser considerada entregue, e sai do pipeline do projeto.

Quem decide pelo cancelamento de uma feature é, unica e exclusivamente, o PO.
Embora a equipe pode achar que a feature seja inimaginavelmente complexa ou irreal, isso não importa, pois o PO pode assim mesmo a desejar e disposto a esperar e pagar por isso. Assim, cabe somente ao PO cancelar uma feature.

O ScrumMaster, o Time e seus Chief Programmers podem, porém, persuadiar o PO para que ele cancele features que não estejam em sintonia com o projeto.

Uma feature cancelada tambem pode voltar para o pipeline do projeto, a criterio do PO. Nesse caso, ela volta e inicia no status Desejada.

3. Tipos

> **Atualizacao Em Progresso**
>
> Esta secao esta sendo atualizada.
> Para cada tipo de feature, um determinado fluxo de trabalho (workflow) está associado.

3.1. Tabela de Features

Representacao	Nome	Descricao
	**User**	Representa uma característica ou função de alto de valor agregado para o cliente que eh executada pelo proprio usuario final do sistema, tais como uma tela de sistema, uma operação em uma tela, etc.
	**Develo per**	Atividade desempenhada por membro da equipe do projeto, que visa agregar valor ao projeto durante a sua execucao, e nao apos a sua conclusão, tais como mentoria, consultoria, treinamento, instalações, etc.

	Acquisition	Semelhante à Developer Feature, é uma atividade de aquisição de recursos para o projeto desempenhada por membro da equipe, e que visa agregar valor ao projeto durante ou após sua conclusão. Esse tipo de feature existe para satisfazer o Processo de Aquisições, descrito pelo MPS.br.
	System	Uma função ou característica que deve ser suportada e executada pelo próprio sistema final, sem participação de um usuário, tais como processos de integracao automatizados. Também contempla as "habilidades" (requisitos nao funcionais) do sistema.
	Overload	Tarefa desempenhada por membro da equipe do projeto que nao eh percebida diretamente pelo cliente final, mas que eh essencial para a conducao do projeto, tais como gerenciamento, infraestrutura, suporte tecnologico, pesquisas, reunioes de equipe, etc. Para mais detalhes, veja item relacionado "Overload Task" em http://xprocess.blogspot.com/2009/05/what-is-overhead-task.html
	Fail	Atividades que representam defeitos detectados antes da entrega da feature. São os únicos sub-tipos dentro do processo 3PUP, e podem existir ou serem criados em qualquer momento (status) de uma feature não finalizada (entregue) ainda. A existência de falhas não finalizadas, implica que a feature não pode ser finalizada também.
	Risk	Qualquer condição mapeada, ou seja, prevista, que, se ocorrer, pode afetar o projeto de alguma forma, negativa ou positivamente. Os riscos são calculados utilizando abordagem do PMBOK, onde a sua prioridade é o equivalente a multiplicação dos Feature Points (FEP) versus a probabilidade da sua ocorrência. É o inverso do happen.
	Happen	Qualquer evento não mapeado, ou seja, imprevisto, que atue de forma negativa ou positivamente para o projeto, comprometendo ou melhorando seu desempenho. É o inverso do risco.
	Epic	Representa um conjunto de features relacionadas. Um épico é utilizado para agrupar features distintas que têm um objetivo comum. Pense no épico como o Ciclo de Vida de um objeto, sendo cada feature uma das partes desse ciclo (criar, editar, deletar, pesquisar, exportar, importar, etc.)
	Set	Representa um conjunto de épicos fortemente relacionados.
	Theme	Representa um conjunto de sets.

3.2. Exemplos de Features

Tipo	ATOR	AÇÃO	artigo	RESULTADO	preposição	OBJETO DE DOMÍNIO	preposição	OBJETIVO DE NEGÓCIO
	Operador de terminal	imprimir	o	espelho	da	nota fiscal eletrônica	para	permitir liberação do transporte
	Administrador	listar	os	usuários ativos	do	sistema	para	verificar quem está contando como licenciamento
	Cliente	adicionar	o	item de compra	ao	carrio de compra	para	avançar na sua compra
	Administrador	criar	o	modelo arquitetural	do	sistema	para	para nortear estrutura-base de desenvolvimento.
	Equipe	realizar	a	reunião de lançamento	do	projeto	para	comunicar o plano de projeto a todos
	Gerente	gerenciar	as	atividades	do	projeto	para	manter cronograma de trabalho do projeto

4. Progresso

> ⚠ Esta seção não está totalmente aderente ao processo 3PUP, onde os índices de progresso listados representam uma simplificação do processo. Isso deve ser revisado quando a documentação completa for refeita.

Conforme o o fluxo da feature avança do início ao fim, o percentual de progresso dela evolui conforme tabela abaixo:

Status	Progresso (%)
Desejado	1
Priorizado	2
Projetado, Rejeitado	5
Planejado	7

Construção, Parado	50
Aceite	90
Entregue, Cancelado	100

5. Tamanho

O tamanho de uma feature corresponde ao número de horas-homem necessárias para sua construção, desde de a Definição até a Entrega.

As features utilizam a técnica "dobro do dobro" para definição do tamanho.

Uma feature precisa estar enquadrada em um dos seguintes tamanhos:

- 2h : Menor tamanho possível = meio turno
- 4h : Um turno
- 8h : Um dia
- 16h : Dois dias
- 32h : Maior tamanho possivel = uma semana útil (5 dias uteis, medido à 80% de produtividade)

TODO: Mais informações sobre "o porquê" dessa técnica e valores devem ser acrescidos em breve.

6. Artefato

Toda feature visa produzir um determinado tipo de arefato relacionado, seja ele uma tela de sistema, um processo de integracao, um algoritimo, um documento, etc. Por exemplo, a User Feature "Usuario manter cadastro de enderecos" produz o artefato UI Simple.

Abaixo, os tipos de artefatos que podem ser produzidos por uma feature:

Artefato	Descricao	
UI Simple	Uma interface de usuario generica simples, que nao esta vinculada a operacoes de edicao de dados no banco de dados. Pode ser uma tela principal de um sistema, uma tela de dialogo, uma tela de importacao de dados, uma tela para envio de email, uma tela de configuracao de alguma coisa. Diz-se simples quando o foco da tela esta em somente mostrar um conjunto unificado de informacoes, como um menu, uma arvore ou um painel de dados.	
UI Composite	Segue as mesmas bases da UI Simple, com o diferencial que esta tela eh formada pela agregacao de diversos paineis com dados que podem estar vindo de outras telas. Como exemplo, pense um uma tela onde aparece um menu no lado esquerdo e um painel centralizado com informacoes de um item de menu. Ou uma tela onde sao abertos varios formularios um ao lado ou um acima do outro, ou mesmo uma tela onde sao usadas abas para visualizacao de multiplas informacoes	
UI List	Uma tela para listagem de dados provenientes do banco de dados, incluindo ou nao suporte para filtro de informacoes. Pode ou nao conter regras de ordenamento, somatorio, etc. Tambem pode incluir capacidade para edicao inlide de dados na propria listagem	
UI Filter	Uma tela para filtro de informacoes, como uma tela de pesquisa (tela do Google por exemplo) ou uma tela de filtragem para visualizacao de dados em um relatorio. Tambem pode ser uma parte de uma tela, como no Gmail onde existe um "box" para digitar filtros de emails, ou ma tela de "lookup" de registros ou mesmo uma tela de filtro e edicao como a que existe no Eclipse IDE, Word ou Notepad++	...
UI CRUD Simple	CRUDs sao telas para manutencao (inclusao, edicao, visualizacao e exclusao) de dados do banco de dados. Uma CRUD simples atua somente sobre um objeto do banco de dados (uma tabela), podendo ou nao ter referencias de uso para outras tabelas (ex.: uma tela de cadastro que contenha comboboxes com valores oriundos de outras tabelas no sistema). Pode ou nao ter regras de validacao ou negocio associadas ao salvamento/carregamento das informacoes.	
UI CRUD Detail	Analoga a UI CRUD simples, eh uma tela CRUD que pode ser acessada diretamente pelo usuario, mas que eh dependente de um registro principal. Tambem pode ser acessada a partir de uma tela mestre-detalhe. Em qualquer caso, a tela CRUD Detail sempre contera uma referencia para um registro principal (o master), embora este possa ser nulo (conceito de detalhe independente). Se o registro principal for obrigatorio, entao diz-se que eh uma CRUD Detail depentente. CRUDs Detail dependente geralmente tem um comportamento de delecao em cascada junto com a delecao do registro principal, ja CRUDs Detail independentes geralmente sobrevivem a delecao do registro pai, onde a referencia para ele entao torna-se nula.	
UI CRUD Master-Detail	Uma tela CRUD que mantem a edicao dos dados do registro principal relacionado, mas que tambem inclui comportamentos para adicionar, listar, excluir informacoes de registros dependentes dela. Sao telas complexas por natureza. Podem ainda estar relacionadas umas com as outras diretamente, formando o conceito de telas com dependencia muitos-para-muitos (ex.: um cadastro de cursos - que contem disciplinas vinculado a um cadastro de alunos - que podem diversos cursos, e que estao matriculados em diversas disciplinas)	...
Report List	Semelhante a uma tela UI List, permite uma saida alem do monitor grafico padrao do sistema, incluindo suporte para exportacao de dados em diversos formatos.	
Report Aggregate	Estende o Report List padrao incluindo capacidades para funcoes de soma, media, agregacoes, quebras, agrupamentos e outras operacoes sobre os dados exibidos.	

System function	Funcionalidade que nao possui uma saida grafica para o usuario final, e apenas eh realizada pelo proprio sistema diretamente, como processos de integracao, schedulers, geracao de relatorios em background, processamento de filas de mensagens assincronas, e as "habilidades" do sistema, como seguranca, performance, usabilidade, etc.
User Document	O resultado da feature eh um documento em formato qualquer, como wiki, .doc, .ppt ou qualquer coisa. Geralmente, features que sao do tipo Developer Feature podem produzir esse tipo de arefato, como por exemplo a criacao de uma apostila para um treinamento ou a documentacao de uma funcionalidade do sistema, como um guia do usuario, diagrama de arquietura, etc.
User Process	O resultado da featura eh um processo de negocio, como um workflow ou atividade. Features do tipo Overload Feature geralmente produzem esse tipo de saida, como uma reuniao de planejamento, uma interacao de modelagem de dominio, a participacao em algum evento, atividades de pesquisa, etc.

7. Nomenclatura

Para ser válida, o nome de uma feature precisa seguir o padrão *AARON*, onde:

- A: Ator
- A: Ação
- R: Resultado
- O: Objeto
- N: objetivo de Negocio

Exemplo:

Operador de terminal (o Ator) imprimir (a Ação) o espelho (o Resultado) da nota fiscal eletrônica (o Objeto) para permitir liberação de transporte (o Motivo) _.

TODO: Mais informações sobre "o porquê" dessa técnica deve ser acrescido em breve.

8. Conteúdo

O conteúdo de uma feature (sua descrição interna e requisitos) variam conforme o status no ciclo de vida.

Assim, para cada etapa do ciclo de vida, um conteúdo distinto deve estar disponível, conforme:

- Para cada status à direita, subentende-se que os valores à esquerda já estão preenchidos.

0 - Desejada	1 - Definida	2 - Planejada	3 - Projetada	4 - Construida	5 - Validada	6 - Entregue	7 - Cancelada
Nome	Local, Descricao, Gestor, Requisitos	Tamanho, Versao, Prioridade	Caso de uso (nome, objetivo, atores, eventos de disparo, pre-condicoes, fluxo principal, fluxos alternativos, pos-condicoes, excecoes), Dominio, Interface, Testes definidos, Implantacao	Artefatos implementados, Testes realizados	Testes aceitos	Aceite	Motivo, Aceite

9. Responsabilidades

Os seguintes perfis estao relacionados a uma feature:

1. Interessado: É o primeiro "A" do nome da feature. Geralmente, é o usuário final.
2. Representante: É quem levantou o desejo da feature no projeto. Geralmente, o Product Owner.
3. Gestor: É o membro da equipe de desenvolvimento do projeto responsável pela feature durante todo o seu ciclo de vida. Geralmente, um projetista.

10. Bons Exemplos

TODO.

11. Maus Exemplos

TODO.

12. Mais Informações