

eBook

REQUISITOS DE SOFTWARE

Uma visão detalhada sobre Requisitos
Funcionais, Requisitos Não-Funcionais e Regras
de Negócio

Uma produção do blog de Engenharia de Software:
<http://www.ateomomento.com.br>

Indtech



Eu sou o Plínio Ventura, do blog [Até o Momento](http://www.ateomomento.com.br) e da [Indtech Academia de Software](http://www.indtech.com.br).

É um prazer poder lhe oferecer este eBook!

Espero realmente que este material seja muito útil a no seu dia a dia profissional.

Trabalho com TI desde 1998, e pude viver um pouco de tudo na área de Engenharia de Software.

Já atuei como Programador, Analista de Sistemas, Coordenador de Qualidade, Coordenador de Desenvolvimento, empreendedor em startup e Gerente de Projetos.

Em todo este tempo, não restou dúvida sobre o que mais gosto: gosto de ser técnico! As coisas podem e devem ser bem-feitas; realmente não acredito que política e pressão realizam bons projetos. Acredito na qualidade e no compromisso, na excelência técnica e no time horizontal!

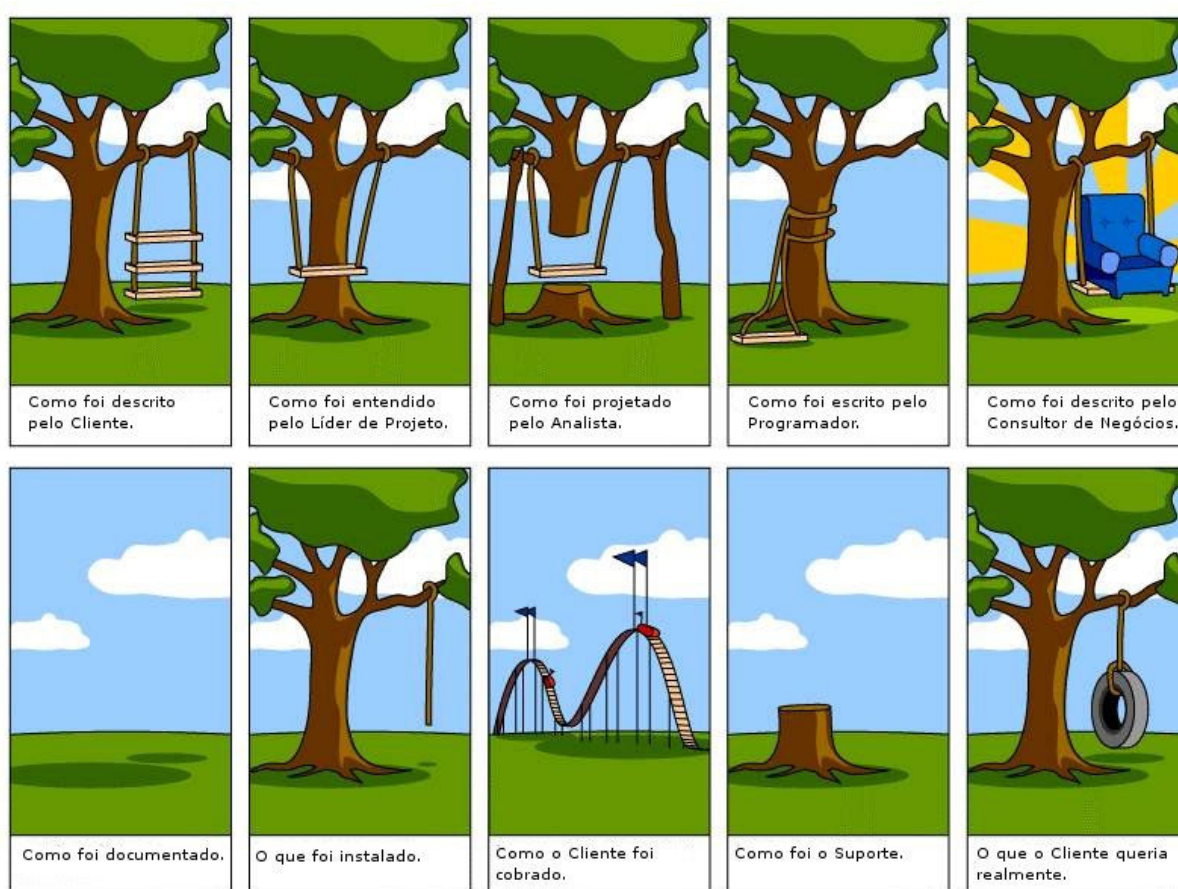
Nestes longos anos sempre fiquei muito (muito mesmo) incomodado com a forma como as empresas lidavam/lidam com Engenharia de Software, principalmente quando o assunto era Engenharia de Requisitos.

Requisitos são o início de tudo, são a causa de “todo o resto” em projetos de software. Entendê-los corretamente, e materializar este entendimento no dia a dia, faz toda a diferença no sucesso dos projetos.

Aqui tem um conteúdo bacana sobre isso, e que será muito útil a você.

Grande abraço, bons estudos, e muita energia positiva!

Indtech Academia de Software
<http://www.indtech.com.br>
<http://www.ateomomento.com.br>
<http://www.facebook.com/ateomomento>
<https://www.youtube.com/c/plinioventura>



Essa imagem (não sei o autor da que inseri acima, mas já é de domínio público por ter inúmeras versões e variantes feitas por diversas pessoas) reflete perfeitamente a nossa realidade nos projetos de software. E tudo começa nos Requisitos...

Requisitos de Software

Achar uma definição objetiva e exata para a palavra “requisito” é difícil. A palavra possui alguns significados (conforme os dicionários), mas chega a ser um pouco abstrata. Mas basicamente, quando falamos de requisito, estamos falando de **necessidade**, **exigência**, desejo, solicitação. Levando esta palavra para o contexto de um software, estamos falando de **necessidades de um usuário**, **exigências do negócio**, desejos da empresa, solicitação da empresa, tudo isso devendo ser realizado por um sistema, ou seja, o software deverá atender estas necessidades, exigências, desejos e solicitações, e materializar isso em um sistema.

Acho legal pensarmos no software como uma caixa de ferramentas, onde cada ferramenta contida dentro desta caixa é uma funcionalidade, que atende um ou mais requisitos do sistema.



No escopo de um software é comum se ter muitos requisitos, e por uma questão de método, estes requisitos são **agrupados e trabalhados conforme seus objetivos**. Entendemos que requisitos possuem um objetivo só, que é atender a uma exigência informada por alguém, mas a natureza de tal exigência pode ser diferente para cada requisito. Em função disso separamos e trabalhamos os requisitos conforme a natureza de cada um, conforme **o tipo de cada necessidade**, o **tipo** de cada requisito.

Na Engenharia de Software existe um hábito de se categorizar demais, de se classificar demais, e isso torna os métodos/processos aplicados muito inchados, com muitas coisas para fazer (em termos de documentação por

exemplo), coisas que nem sempre geram valor na produção final de um sistema.

Isso ainda é uma realidade, infelizmente, mas percebo que é uma realidade que tem ficado mais no campo da teoria, não mais indo para a prática como acontecia nas décadas pretéritas.

A Engenharia de Software é uma ciência/área de conhecimento relativamente nova. Tudo começou formalmente na década de 1960, então estamos falando de uma área de conhecimento com menos de 70 anos. Áreas como a engenharia civil já existem há mais de 3000 anos, apenas para comparação. Qualquer área de conhecimento que surge começa no caos até que se conheça melhor o todo e suas partes, para daí se buscar sair do caos e ir para a ordem.

Mas quando essa busca pela ordem começa, todos querem organizar tudo, e geralmente utilizam-se métodos e processos para isso, baseando-se consciente/inconscientemente no método científico¹.

E neste momento é natural o exagero na burocracia. Tempos depois, chega-se ao equilíbrio. Estamos vendo isso na Engenharia de Software hoje com os métodos ágeis, que pregam menos burocracia e mais software executável para geração de valor, muito diferente do que tínhamos na década de 90, que era a tentativa de organizar o caos que se criou nas décadas de 70/80/90 a um custo proibitivo em termos de burocracia.

Algumas literaturas definem diversas classificações/tipos para requisitos: requisitos de sistema, requisitos de software, requisitos emergentes, requisitos de produto, requisitos de projeto, requisitos de processo, requisitos de teste etc. E na prática, no dia a dia nas fábricas de software, nos projetos e nas operações das empresas, fica claro que quanto menos “denominações” existirem mais simples ficam as coisas e mais objetivas e rápidas também.

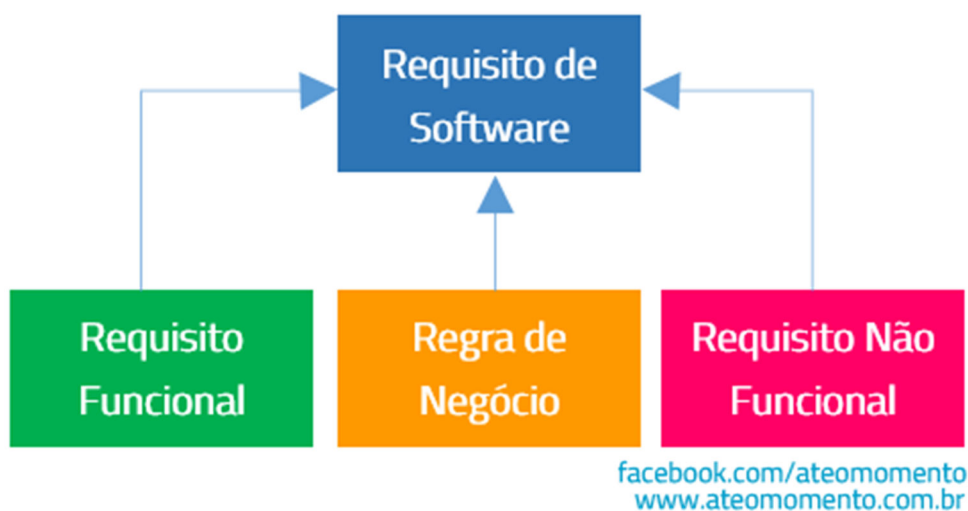
Eu defendo e acredito que em projetos de software é mais do que necessário haver apenas **três tipos** de requisito: **Requisitos Funcionais**, **Requisitos Não Funcionais** e **Regras de Negócio**.

1 https://pt.wikipedia.org/wiki/Método_científico

Na literatura de engenharia de software, de um modo geral, regras de negócio não são tratadas como requisitos, em vários casos nem são mencionadas. Mas eu as coloco no mesmo nível de importância que os requisitos funcionais e Não Funcionais, pois sem elas não existe sistema, sem elas não existem requisitos funcionais.

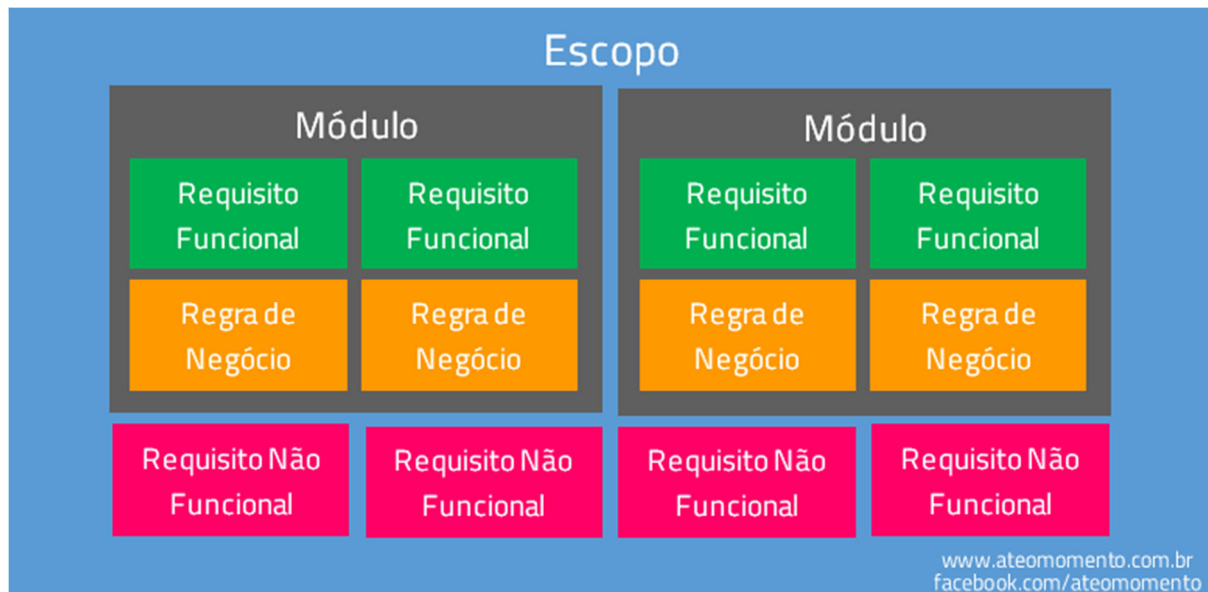
Essa visão se aplica por dois motivos principais: é o que se pratica efetivamente no mercado, e mais do que isso, gera-se burocracia além do permitido para projetos de software (que já requerem um mínimo de burocracia).

Abaixo segue a visão que citei anteriormente, para requisitos de software e seus elementos:



Na imagem acima, o que temos é uma especialização de requisitos de software. O “requisito de software” é o tipo mais genérico (generalização), e “Requisito Funcional”, “Regra de Negócio” e “Requisito Não-Funcional” são tipos mais especializados (especialização).

Conceitualmente falando, um sistema nada mais é que: um **escopo**, dividido em **módulos**, cada módulo com seus **requisitos funcionais** e **regras de negócio**, e **requisitos Não Funcionais** fora dos módulos, permeando todo o sistema. Abaixo uma figura ilustrando um pouco dessa decomposição das partes, no todo que é o software.



Vamos detalhar cada um dos três tipos de requisitos e entender melhor do que se trata cada um deles.

O nível de detalhe nas especificações é pré-requisito para um projeto de software seguro, blindado, com qualidade. Mas isso não significa que devemos criar complexidade quando podemos ter simplicidade.

Simplicidade é fundamental nos projetos de software.



**MENOS É MAIS.
ABAIXO A
COMPLEXIDADE
DESNECESSÁRIA.
SOFTWARE BOM
É SOFTWARE
SIMPLES!**

Mais simplicidade,
menos bugs!

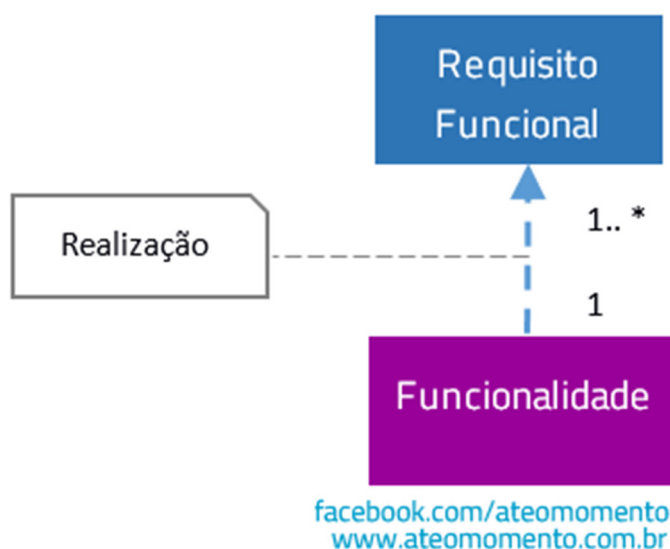
www.ateomomento.com.br
facebook.com/ateomomento

Requisito Funcional

Como vimos no capítulo anterior, requisito é uma **exigência**, solicitação, desejo, **necessidade**. Quando falamos de um Requisito Funcional estamos nos referindo à requisição de uma funcionalidade que um software deverá atender/realizar. Ou seja, exigência, solicitação, desejo, necessidade, que um software deverá materializar.

É comum os profissionais de engenharia de software associarem a ideia de um Requisito Funcional a uma tela, uma rotina, que no fim serão as funcionalidades de fato de um sistema. Mas é necessário entender que **uma funcionalidade não necessariamente realizará apenas um Requisito Funcional**, podendo realizar **vários** requisitos funcionais – significa que em uma funcionalidade um ou mais requisitos funcionais podem ser atendidos, não necessariamente apenas um. Se pensarmos em multiplicidade², uma funcionalidade pode realizar um ou muitos requisitos funcionais (1.. *).

A figura abaixo ilustra esta relação.



Para entender melhor isso vamos a um exemplo mais básico. Imaginemos um sistema que possui uma tela para “Manutenção de Clientes”, que mantém os dados cadastrais de um cliente no sistema. Estamos falando de uma **única funcionalidade**. Nesta tela é possível

²

http://www.ibm.com/support/knowledgecenter/SS5JSH_9.5.0/com.ibm.xtools.petal.ui.doc/topics/rkeydifmultiplicity.html?lang=pt-br

incluir/alterar/consultar/excluir clientes dos tipos pessoa física e pessoa jurídica. Mas quantos requisitos são realizados (atendidos) por esta funcionalidade? **Oito requisitos**. Vejamos a lista a seguir:

Requisitos Funcionais (Identificador e Nome)
RF001 – Incluir cliente pessoa física
RF002 – Alterar cliente pessoa física
RF003 – Consultar cliente pessoa física
RF004 – Excluir cliente pessoa física
RF005 – Incluir cliente pessoa jurídica
RF006 – Alterar cliente pessoa jurídica
RF007 – Consultar cliente pessoa jurídica
RF008 – Excluir cliente pessoa jurídica

*O que **não** é um Requisito Funcional?*

É comum quando se fala de Requisito Funcional associar a isto funcionalidade, caso de uso, Regra de Negócio ou até mesmo requisito não funcional. São coisas muito diferentes.

Uma funcionalidade pode realizar um ou mais requisitos funcionais. Requisito funcional não é uma funcionalidade, é uma necessidade funcional (uma função) que o software deve atender. Uma funcionalidade será executada por um ator (um ator sistêmico [pelo próprio sistema] ou um ator humano [usuário final]). É onde requisitos funcionais serão viabilizados.

*Um caso de uso³ é uma especificação do **comportamento** de uma funcionalidade. Nele se tem detalhes sobre como a funcionalidade “funcionará”, com restrições, premissas e diretrizes pertinentes à funcionalidade.*

Regra de negócio refere-se a premissas ou restrições de negócio que o sistema deverá atender, regras que poderão ou não estar associadas a um Requisito Funcional, mas que sempre serão realizadas por uma ou

³ <http://www.ateomomento.com.br/o-que-e-caso-de-uso/>

*mais funcionalidades do sistema. Na visão da **modelagem conceitual**, Regras de negócio são o “como”, requisitos funcionais são o “o que”.*

Requisitos Não Funcionais são premissas ou restrições que o sistema deverá atender, mas que não são realizados através de funcionalidades. Podem ou não estar associados a requisitos funcionais, mas não tem, necessariamente, relação com o negócio, na visão do usuário.

Importância dos Requisitos Funcionais

Funcionalidades somente existem para realizar requisitos funcionais. Logo, sem requisitos funcionais não há funcionalidades e sem funcionalidades não há sistema. Este raciocínio por si só demonstra a importância absoluta e inquestionável dos requisitos funcionais no escopo de um sistema.

E por ser algo importante como é, todo cuidado é pouco para que estes requisitos possuam a maior qualidade possível, pois apenas a existência deles no escopo não garante um bom sistema, eles precisam ter qualidade em termos de sintaxe e semântica⁴. Precisam ser bem feitos. Mas o que devemos entender como qualidade de um requisito?

Atributos de um bom Requisito Funcional

Um Requisito Funcional de qualidade precisa atender alguns atributos específicos. Na literatura, principalmente estrangeira, existem várias recomendações de atributos que um requisito deve atender para ter qualidade. Mas vou me ater apenas aos que realmente considero relevantes na prática, que fazem diferença no dia a dia. A seguir a lista dos atributos que considero relevantes.

Atributo	Referente a
Unidade	O RF deve propor uma única coisa apenas. Não deve atender a mais de uma exigência. O RF “Incluir cliente” não é unitário, pois se refere a incluir clientes de tipos diferentes (pessoa física e jurídica), assumindo assim várias responsabilidades, quando deveria assumir apenas uma.
Compleitude	O RF deve ser autocontido, deve ter “início/meio/fim”, ser completo. O RF “Pagar fatura” não é completo, só conta “parte

⁴ <http://www.ateomomento.com.br/sintaxe-semantica-software/>

Atributo	Referente a
	da estória”. Para ser completo deveria ser algo como “Pagar fatura com cartão de crédito para cliente pessoa física”.
Consistência	O RF não deve contradizer outro RF do mesmo escopo do projeto. É como termos dois RFs se propondo a fazer uma mesma coisa, mas cada RF se propondo a fazer esta coisa de uma forma diferente.
Atomicidade	Um RF para ser atômico precisa também ter unidade, pois atomicidade remete a assumir apenas uma responsabilidade. Mas também deve ser algo indivisível, não podendo ser decomposto. Muitos RFs possuem conjunção, dependem de outros para se realizarem. Onde temos dois RFs “Realizar compra de produto” e “Realizar pagamento com cartão de crédito” na realidade, se pensarmos em atomicidade, temos um único RF que é “Realizar compra de produto com pagamento em cartão de crédito”.
Não-Ambiguidade	Um RF não pode ser ambíguo, não pode propor algo que não fica claro o que é. O RF “Emitir relatório” não quer dizer nada. Relatório de que, para que? “Emitir relatório de saldo” já é melhor, mas ainda é ruim. Saldo de que? Seria não ambíguo se não deixasse dúvidas, algo como “Emitir relatório de saldo da conta corrente do cliente pessoa física”.
Verificável	Não adianta ter um RF se ele não é palpável, possível de associar com um artefato de construção, de testes. Um RF tem que ser testável, tem que ser possível atestar que o RF foi atendido, foi construído, foi homologado. Para isso tem que ser também rastreável.
Rastreável	Deve ser possível achar o RF no sistema pronto, funcional e executável. Como saber se um RF foi atendido? Para isso é necessário ter rastreabilidade, e isso só é possível ligando as pontas (associar o RF à interface gráfica, que será associada a um caso de uso, que será associado a funcionalidades, que serão implementadas etc.).
Prioridade ⁵	Um RF Essencial é algo muito diferente de um RF Desejável, possuem valores para o negócio completamente diferentes.

⁵ <http://www.ateomomento.com.br/priorizacao-de-requisitos/>

Atributo	Referente a
	O RF deve possuir sua prioridade, isso interfere diretamente no projeto do software.

Um detalhe fundamental é o uso do tempo verbal no nome do RF. Um RF, em tempo de especificação, **refere-se a algo que será feito, uma ação a ser realizada pelo sistema**. Por isso o nome precisa estar no tempo verbal infinitivo. Um RF que fala sobre “expurgo de registros de clientes inativos” não pode ter este nome, deve se chamar “Expurgar registros de clientes inativos”. É uma necessidade, mas que precisa ser verbalizada como uma ação a ser realizada.

Estrutura de um Requisito Funcional

Não há um padrão estabelecido sobre a estrutura de um RF. Mas a maioria das empresas utiliza um formato semelhante, contendo campos específicos. O modelo a seguir contempla os campos mais relevantes, com posterior descrição de cada um.

O modelo citado é aplicável em especificações produzidas em Editores de Texto como o Microsoft Word, por exemplo. É recomendado que se utilize, sempre que possível, alguma ferramenta Case⁶ que dê suporte a Engenharia de Requisitos, para melhorar a produtividade na modelagem e a gestão dos requisitos.

Identificador	<<Numero>>		
Nome	<<Texto>>		
Módulo	<<Texto>>		
Data de criação	<<Data>>	Autor	<<Texto>>
Data da última alteração	<<Data>>	Autor	<<Texto>>
Versão	<<Numero>>	Prioridade	<<Texto>>
Descrição	<<Texto>>		

⁶ https://pt.wikipedia.org/wiki/Ferramenta_CASE

Campo	Descrição
Identificador	Sufixo seguido de um identificador único. O sufixo geralmente utilizado é RF (Requisito Funcional) e o identificador único geralmente é composto de quatro dígitos.
Nome	Nome curto do RF, mas que possibilite entender bem o que RF faz apenas pelo nome.
Módulo	Módulo ao qual o RF pertence. Se for um sistema pequeno que não possua nenhum módulo, somente o próprio sistema, deve ser preenchido com N/A (não se aplica).
Data de criação	Data da criação do RF, ou a data em que ele foi especificado.
Autor	Profissional que especificou o RF pela primeira vez, quem o criou.
Data da última alteração	Data em que houve a última alteração no RF.
Autor	Profissional que alterou a especificação do RF pela última vez.
Versão	Número da versão do RF. Geralmente utiliza-se algo simples, como 1, 2. A versão inicial sempre é a 1, e a cada alteração incrementa-se a versão (na criação versão 1, na primeira alteração versão 2 e por aí vai).
Prioridade ⁷	Se o RF é Essencial, Importante ou Desejável.
Descrição	Descrição detalhada (a mais detalhada possível) do RF.

O uso de identificador é fundamental para uma melhor organização do projeto. Localizar requisitos pelo nome não funciona, devido ao volume dos requisitos e ambiguidades nos nomes. E também, é importante ter algum recurso de numeração automática, pois controlar os números dos identificadores manualmente é inviável.

Exemplos de requisitos funcionais especificados

Vejamos alguns exemplos de requisitos funcionais especificados:

⁷ <http://www.ateomomento.com.br/priorizacao-de-requisitos/>

Identificador	RF0001		
Nome	Consultar automaticamente o CEP de clientes através do endereço residencial		
Módulo	Cadastro		
Data de criação	31/01/2016	Autor	Arquitas de Tarento
Data da última alteração	N/A	Autor	N/A
Versão	1	Prioridade	Desejável
Descrição	<p>Para todos os clientes cadastrados deverá ser possível consultar o CEP do endereço residencial de maneira automática através dos dados de logradouro, número, complemento, bairro, cidade e estado.</p> <p>A consulta se dará após os dados citados serem informados, de maneira transparente para o usuário final. Não deve ser necessário clicar em botão ou acionar algum outro comando para que a consulta ocorra.</p>		

Identificador	RF0002		
Nome	Herdar permissões de acesso para o usuário que for associado a um grupo de usuários previamente cadastrado e ativo		
Módulo	Segurança		
Data de criação	31/01/2016	Autor	Nelson Goodman
Data da última alteração	N/A	Autor	N/A
Versão	1	Prioridade	Essencial
Descrição	<p>O sistema trabalha com o conceito de grupo de usuários, onde todo usuário deve fazer parte de, ao menos, um grupo ativo (grupos inativos não se aplicam).</p> <p>Para informação, não é permitido atribuição individual de permissão diretamente ao usuário. As permissões se dão</p>		

	<p>apenas no nível de grupo. Cada grupo de usuários possui um perfil pré-configurado de permissões de acesso.</p> <p>Sempre que um usuário for associado a um grupo, este usuário deve herdar as permissões de acesso pré-configuradas para o grupo em questão, e mantê-las até enquanto fizer parte do grupo.</p>
--	--

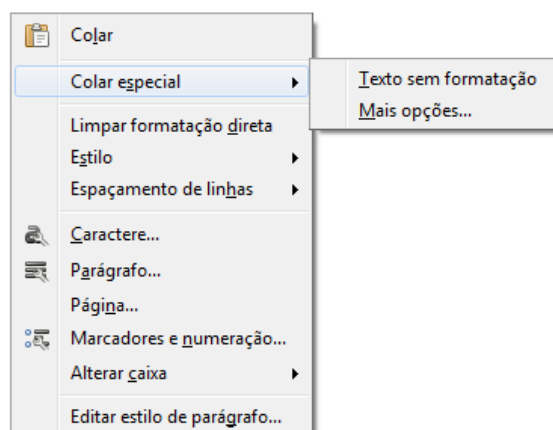
Identificador	RF0003		
Nome	Emitir carta de cobrança para clientes inadimplentes conforme critérios pré-estabelecidos		
Módulo	Cobrança		
Data de criação	31/01/2016	Autor	Hípias de Elis
Data da última alteração	N/A	Autor	N/A
Versão	1	Prioridade	Essencial
Descrição	<p>Para todo cliente que esteja inadimplente deverá ser possível a emissão de carta de cobrança através do sistema. O resultado da emissão será a carta impressa, que será posteriormente despachada por correios ou outro agente de entrega.</p> <p>Os critérios para definir se um cliente é ou não inadimplente devem estar cadastrados no sistema utilizando regras de negócio específicas. Nem todo cliente com pagamento em atraso será considerado inadimplente, pois deverá ser levado em consideração o score de crédito do cliente, sua renda mensal, patrimônio etc. Obs.: verificar os requisitos funcionais e regras de negócio específicas para manutenção dos critérios citados.</p> <p>Não haverá diferença na emissão da carta de cobrança para clientes pessoa física (particular) ou jurídica (empresarial), logo, o processo será o mesmo para ambos.</p>		

Identificador	RF0004		
Nome	Recalcular contribuições calculadas incorretamente que não tenham sido pagas para participantes ativos		
Módulo	Contribuições		
Data de criação	31/01/2016	Autor	Lucrécio
Data da última alteração	N/A	Autor	N/A
Versão	1	Prioridade	Importante
Descrição	<p>Após o cálculo das contribuições de um participante (participante de qualquer plano previdenciário cadastrado no sistema) haverá a conferência manual, por um atuário, dos valores de contribuição que foram calculados pelo sistema.</p> <p>Para cada contribuição que o atuário detectar que houve erro no cálculo e que ainda não foi paga pelo participante, poderá haver o recálculo através do sistema (o sistema deverá ter um recurso para recálculo da contribuição). O recálculo deverá ser possível para uma única referência (mês/ano) e um único participante; para vários participantes numa mesma referência; para um único participante em várias referências (considerando um período – referência inicial e final) e para vários participantes em várias referências.</p> <p>Para recálculo de contribuições calculadas incorretamente, mas que já foram pagas, verificar RF correspondente. Neste caso será necessário estornar o pagamento antes de efetuar o recálculo da contribuição.</p>		

Exemplos de requisitos funcionais materializados

Vimos requisitos funcionais especificados. Isso ocorre em tempo de especificação dos requisitos, na fase do ciclo de vida do projeto em que a modelagem de requisitos acontece.

Quando o sistema está pronto, os requisitos são realizados por funcionalidades do sistema. Vejamos a seguir exemplos de requisitos funcionais materializados, ou seja, implementados.



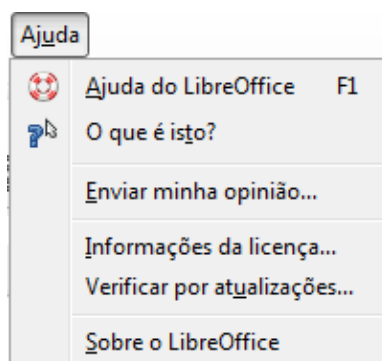
Isto é um menu de contexto, o famoso menu que é exibido após o clique com o botão direito do mouse (pode ser o esquerdo também, dependendo da configuração no sistema operacional). Neste menu temos diversos comandos que podem ser executados, onde podemos extrair facilmente mais de uma dezena de requisitos funcionais que nesta funcionalidade (menu de contexto) são realizados.

Quando o comando **Colar especial** com a opção **Texto sem formatação** é executado o conteúdo que está armazenado na área de transferência é colado no corpo do texto, assumindo a formatação padrão do local do documento onde está sendo feita a “colagem”.

Vejamos como seria a especificação deste Requisito Funcional.

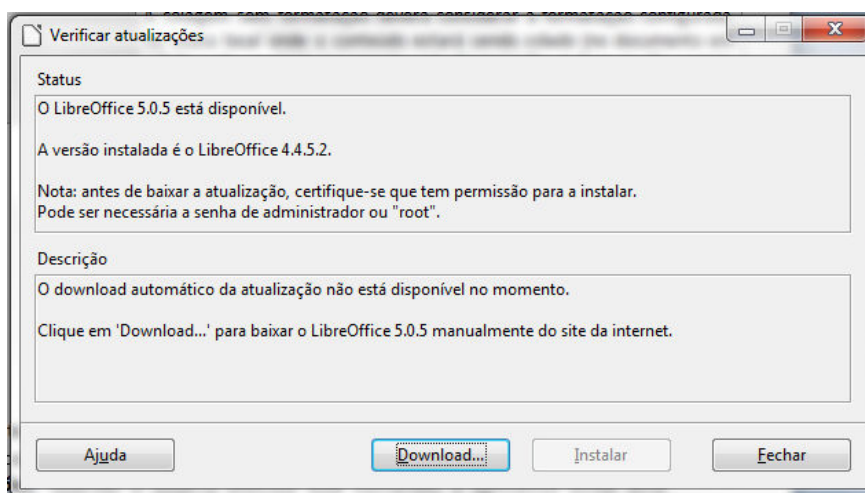
Identificador	RF0019		
Nome	Colar sem formatação o texto copiado da área de transferência		
Módulo	Editor de Texto		
Data de criação	25/02/2016	Autor	Eckhart
Data da última alteração	N/A	Autor	N/A
Versão	1	Prioridade	Essencial
Descrição	Após a cópia de algum conteúdo textual para área de transferência do sistema operacional, o usuário poderá realizar a colagem deste conteúdo no documento que estiver editando.		

A colagem sem formatação deverá considerar a formatação configurada no exato local onde o conteúdo estará sendo colado (no documento em edição pelo usuário), assumindo esta formatação. A formatação contida na origem onde o texto foi copiado deverá ser ignorada.



Este é um sub menu que é exibido após o clique no item de menu **Ajuda**. Temos diversos comandos no sub menu e vamos analisar o comando **Verificar por atualizações**.

Quando o usuário executa este comando o aplicativo exibe uma janela onde o resultado da verificação da atualização é exibido.



Vejamos como seria a especificação deste Requisito Funcional.

Identificador	RF0087
Nome	Verificar por atualizações disponíveis
Módulo	Editor de Texto

Data de criação	25/02/2016	Autor	Erastóstenes
Data da última alteração	N/A	Autor	N/A
Versão	1	Prioridade	Desejável
Descrição	<p>O usuário deverá ter um recurso para que, sob demanda, verifique se existem atualizações disponíveis para o aplicativo, atualizações que ainda não foram instaladas por ele.</p> <p>Esta verificação deverá ser feita no servidor do fabricante do aplicativo e o aplicativo deverá exibir para o usuário quais as atualizações disponíveis, demonstrando qual a versão instalada na máquina do usuário, e qual a versão atual disponível para atualização.</p>		

Obs.: na janela **Verificar Atualizações** temos vários comandos (Ajuda, Download, Instalar e Fechar). A janela ainda exibe os dados da consulta à versão mais atual disponível para atualização. Esta tela é um exemplo de como uma única funcionalidade realiza vários requisitos. Ao chamar esta tela o aplicativo realiza o Requisito Funcional de verificação de atualizações (RF0087), e através desta mesma tela (funcionalidade) outros requisitos funcionais são realizados, como os de realizar download da versão para atualização e instalar a versão de atualização.

É muito importante refletir sobre a relação de Requisito Funcional e Funcionalidade.

Como já citado, é comum analistas confundirem as duas coisas. E essa confusão leva a prejuízos para o projeto.

O principal prejuízo é o comprometimento do nível de detalhamento dos requisitos funcionais, e o não atendimento aos seus atributos de qualidade. E isso fica pior quando toda a equipe pensa da mesma forma, pois assim assume-se que o que foi produzido na modelagem de requisitos está bom.

Onde se confunde Requisito Funcional e funcionalidade geralmente encontra-se requisitos funcionais como “Realizar Pagamento”, “Emitir

Fatura”, “Transferir dinheiro” e coisas do tipo. Neste nível de detalhe, conflitos entre usuários e analistas ou entre analistas e programadores, por exemplo, é algo fatal.

A seguir, uma janela de preenchimento de perfil de usuário existente numa rede social.

Nesta janela temos um menu lateral que possui diversos grupos de informações. Para cada grupo, temos comandos diversos utilizados no preenchimento das informações.

A imagem mostra a interface de perfil de usuário 'Sobre'. No menu lateral à esquerda, há opções: 'Visão geral', 'Trabalho e educação' (destacado), 'Locais onde você morou', 'Informações básicas e de contato', 'Família e relacionamentos', 'Detalhes sobre você' e 'Acontecimentos'. A seção principal à direita está dividida em quatro sub-grupos: 'TRABALHO' com o botão '+ Adicionar um local de trabalho'; 'HABILIDADES PROFISSIONAIS' com o botão '+ Adicionar uma habilidade profissional'; 'FACULDADE' com o botão '+ Adicionar uma faculdade'; e 'ENSINO MÉDIO' com o botão '+ Adicionar uma instituição de ensino médio'.

No menu “Trabalho e Educação”, temos quatro sub-grupos, cada um com um comando para adicionar informações. Aqui temos quatro requisitos funcionais materializados: Adicionar um local de trabalho, Adicionar uma habilidade profissional, Adicionar uma faculdade, Adicionar uma instituição de ensino médio.

Vejamos como seria a especificação de um destes requisitos:

Identificador	RF0096		
Nome	Adicionar um local de trabalho		
Módulo	Perfil do usuário		
Data de criação	25/03/2016	Autor	Plotino
Data da última alteração	N/A	Autor	N/A
Versão	1	Prioridade	Essencial

Descrição

O usuário poderá inserir informações sobre trabalho e educação em seu perfil na rede social. Neste contexto, deverá poder adicionar seus locais de trabalho (empresas onde trabalhou).

Para cada local de trabalho, deverá poder inserir o nome da empresa, endereço da empresa, data início e data fim do período em que trabalhou na empresa, e a função que exerceu na empresa.

Esta informação poderá ser alterada a qualquer momento pelo usuário, e será exibida aos seus amigos na rede social.

Requisito Não Funcional

Requisito Não Funcional, como o próprio nome diz, é uma “não funcionalidade”, ou seja, trata-se de algo que não é uma funcionalidade, mas que precisa ser realizado para que o software atenda seu propósito.

Existe uma definição propagada na literatura de Engenharia de Software que afirma que um Requisito Funcional define **o que** o sistema fará, e o requisito não funcional define **como** o sistema fará. Alguns afirmam que um requisito não funcional especifica como um Requisito Funcional será implementado; também não é uma boa definição, pois uma funcionalidade teoricamente pode ser implementada sem nenhum requisito não funcional no projeto e isso não gerar ônus nenhum.

Acho que nenhuma dessas definições é suficiente, não dá para entender muito bem; tem outra que diz que requisitos Não Funcionais são **atributos de qualidade para o sistema**, essa eu já acho melhor, mas ainda um pouco subjetiva.

Um RNF tem como objetivo atender a requisitos do sistema que **não são requisitos funcionais** (não se referem a funcionalidades do negócio), mas que fazem parte do escopo do sistema.

Um RNF pode ou não estar associado a um Requisito Funcional (essa associação é muito comum em requisitos Não Funcionais de integração de sistemas, por exemplo, pois para cada integração existe uma necessidade do negócio de fazê-la, necessidade que está descrita em um Requisito Funcional).

Toda necessidade que **for realizada através de funcionalidades** é resultado de um ou mais requisitos funcionais (pois uma funcionalidade pode realizar vários requisitos funcionais, não necessariamente apenas um) e toda necessidade **que não puder ser atendida desta forma**, é um requisito não funcional – geralmente trata-se de premissas e restrições técnicas aplicadas ao projeto. Acho que essa definição, aparentemente simples, define bem um RNF.

Importância dos Requisitos Não Funcionais

Requisitos Não Funcionais são tão importantes quanto os requisitos funcionais ou regras de negócio. Infelizmente é muito difícil encontrar alguma empresa que leve isso a sério, e aí está a causa do fracasso de muitos projetos de software.

Existem duas principais causas que justificam a pouca importância que os RNFs tem na maioria dos projetos:

Usuários não sabem o que é isso

Usuários pensam apenas no negócio, que remete a requisitos funcionais, e conseqüentemente, funcionalidades. Apenas em empresas de grande porte, com uma TI mais estruturada, onde a área de TI apoia os usuários nos projetos é que se dá alguma importância a RNFs;

RNFs são difíceis de estimar

Em termos de volume de trabalho/esforço (horas gastas na produção), geralmente isso é feito baseado na famosa “opinião de especialista”. Medidas como Ponto de Função, Linha de Código, Pontos por Caso de Uso e técnicas semelhantes mensuram apenas o que é perceptível para o usuário, e RNFs geralmente não o são (na última versão da Análise por Pontos de Função há algo mais maduro para medir RNFs, mas na minha opinião ainda não atende por completo). Muitos fornecedores ignoram os RNFs quando não é solicitado explicitamente pelo cliente, pois acham que “gastarão mais” no projeto.

Dependendo de como se enxerga, RNFs podem ser até mais importantes que RFs. Se pensarmos num sistema de grande porte (como um ERP para grandes empresas), no escopo de um sistema como este tem facilmente mais que cinco mil requisitos funcionais e dez mil regras de negócio. Mas podemos ter poucas dezenas de requisitos Não Funcionais. No meio de cinco mil requisitos funcionais, deixar de atender no escopo uns poucos provavelmente não inviabiliza o sistema, no meio de quinze mil regras de negócio idem. Mas existem RNFs que se não são atendidos, inviabilizam um sistema gigante como este.

Diferente dos RFs, os RNFs permeiam todo o sistema (geralmente são “utilizados” em vários módulos do sistema, de maneira horizontal),

existem em quantidade muito menor nos projetos se comparado com os RFs mas possuem uma amplitude enorme no sistema. Um RNF pertinente à usabilidade, por exemplo, algo sobre padrão de barra de rolagem, pode existir em quase todas as telas do sistema.

Para ilustrar, no contexto do ERP citado, vamos imaginar a seguinte necessidade de um cliente, num cenário hipotético:

No módulo de logística, toda carga deve ser liberada em no máximo 20 minutos, pois temos uma frota de dois mil caminhões que não podem esperar mais que isso para sair dos armazéns. Para a liberação da carga, um documento de solicitação de liberação de carga deve ser emitido, e nele anexada a nota fiscal dos produtos contidos no caminhão.

A partir deste cenário imaginamos que um analista identificou um RNF de Desempenho (mais à frente falaremos sobre as categorias dos requisitos Não Funcionais) nomeado de “RNF0087 – Tempo limite para processamento de solicitação de liberação de carga”, onde foi especificada a restrição do tempo de vinte minutos para liberação da carga.

Requisitos Não Funcionais são entradas (input) para a definição da arquitetura técnica de um software. Em sistemas com alto volume de processamento e exigências “desafiadoras” de tempo de resposta, projetar e implementar a arquitetura ignorando os requisitos Não Funcionais é sacramentar o caos vindouro.

Entretanto, os arquitetos responsáveis pela implementação da arquitetura do ERP não deram a devida importância ao RNF citado e pensaram a estrutura do sistema ignorando uma performance otimizada, e esta necessidade de tempo de resposta ficou relegada para o fim do projeto. Quando o produto ficou pronto, iniciou-se a homologação. Na primeira bateria de testes de desempenho o sistema foi reprovado pela área de logística, pois estava demorando 35 minutos, em média, apenas para processar a solicitação de liberação de carga (gerar nota fiscal, anexá-la à solicitação, processar a solicitação e rodar o fluxo no sistema até o caminhão poder sair do armazém). **Efeito:** sistema inviável para produção, necessário reestruturar sua arquitetura porque a performance ficou muito ruim. **Causa:** não atendimento do RNF0087.

Reestruturar uma arquitetura não é uma atividade trivial. E no geral (não é regra) o impacto de uma reestruturação destas é proporcional ao tamanho do sistema. Quanto maior o sistema, mais problemático é realizar alterações arquiteturais.

Já presenciei um caso onde a construção de um sistema levou três anos para ficar pronta, para então iniciar a homologação. Quando começaram os testes, percebeu-se inviabilidade em termos de performance.

Para constar apenas, não precisava chegar ao fim da construção para começar os testes, mas o ciclo de desenvolvimento adotado neste projeto não dava outra opção.

Neste caso, foram necessários mais 10 meses para reestruturação (além do atraso já contraído pelo projeto), orçamento adicional de R\$ 1,2 milhões e desperdício de ao menos R\$ 1 milhão em custo de implementação (horas gastas na implementação “errada” desta parte da arquitetura do sistema). Somando tudo, num sistema de grande porte (4000 pontos de função), prejuízo de R\$ 2,5 milhões.

Estrutura de um Requisito Não Funcional

Como no caso dos requisitos funcionais e regras de negócio, não há um padrão estabelecido sobre a estrutura de um RNF. Mas a maioria das empresas utiliza um formato semelhante, contendo campos específicos. O modelo a seguir contempla os campos mais relevantes, com posterior descrição de cada um.

Como citado para a modelagem de requisitos funcionais, o modelo abaixo é aplicável em especificações produzidas em Editores de Texto como o Microsoft Word, por exemplo. É recomendado que se utilize, sempre que possível, alguma ferramenta Case⁸ que dê suporte a Engenharia de Requisitos, para melhorar a produtividade na modelagem e a gestão dos requisitos.

⁸ https://pt.wikipedia.org/wiki/Ferramenta_CASE

Identificador	<<Numero>>	Categoria	<<Texto>>
Nome	<<Texto>>		
Data de criação	<<Data>>	Autor	<<Texto>>
Data da última alteração	<<Data>>	Autor	<<Texto>>
Versão	<<Numero>>	Prioridade	<<Texto>>
Descrição	<<Texto>>		

Campo	Descrição
Identificador	Sufixo seguido de um identificador único. O sufixo geralmente utilizado é RNF (Requisito Não Funcional) e o identificador único geralmente é composto de dois dígitos (dificilmente um projeto terá mais que 99 RNFs).
Categoria	Categoria à qual o RNF pertence (Desempenho, Usabilidade, Padrão etc.).
Nome	Nome curto do requisito, mas que possibilite entender bem o que RNF faz apenas pelo nome.
Data de criação	Data da criação do RNF, ou a data em que ele foi especificado.
Autor	Profissional que especificou o RNF pela primeira vez, quem o criou.
Data da última alteração	Data em que houve a última alteração no RNF.
Autor	Profissional que alterou a especificação do RNF pela última vez.
Versão	Número da versão do RNF. Geralmente utiliza-se algo simples, como 1, 2. A versão inicial sempre é a 1, e a cada alteração incrementa-se a versão (na criação versão 1, na primeira alteração versão 2 etc.).
Prioridade ⁹	Se o RNF é Essencial, Importante ou Desejável.
Descrição	Descrição detalhada (a mais detalhada possível) do RNF.

⁹ <http://www.ateomomento.com.br/priorizacao-de-requisitos/>

Mais à frente teremos vários exemplos de RNFs, onde poderemos ver o modelo apresentado com todos os campos preenchidos.

Categorias dos Requisitos Não Funcionais

Para uma melhor organização da especificação e semântica do projeto do software, RNFs são separados por categorias, conforme o propósito de cada requisito. A seguir, a lista das principais categorias existentes:

Categoria	Referente a
Desempenho	Desempenho do sistema, restrições de performance, tempo de resposta em processamentos específicos, cargas, velocidade de resposta de processamentos em telas etc.
Disponibilidade	Disponibilidade do sistema em tempo útil, restrições sobre janelas de manutenção, janelas de produção, soluções de contorno quando houver queda de energia etc.
Segurança	Diretrizes pertinentes à segurança do sistema, como algoritmo de criptografia a ser utilizado, regras para criação e manutenção de usuários e senhas, uso de certificados digitais, uso de protocolos seguros específicos, uso de captcha etc.
Interoperabilidade	Necessidades de integração do sistema com outros sistemas, integração com APIs ¹⁰ , componentes, banco de dados externos etc.
Usabilidade	Quantidade máxima de cliques por tipo de funcionalidade, uso de componentes e lógicas de telas específicas, restrição/premissas para uso de componentes gráficos (grids, barras de rolagem, menus), recursos de acessibilidade para deficientes, compatibilidade com idiomas etc.
Compatibilidade	Browser e sistemas operacionais nos quais o software deverá rodar, versões de browser e sistemas operacionais, protocolos compatíveis, versões de linguagens de programação e banco de dados para retrocompatibilidade etc.

¹⁰<http://www.ateomomento.com.br/o-que-e-api/>

Categoria	Referente a
Confiabilidade	Políticas para backup do sistema e seus dados, quantidade limite de erros em cálculos e processamentos com erro, regras para rollback quando houver alguma falha, recursos para restauração automática do sistema em caso de queda de energia etc.
Padrões	Padrões em geral, aplicáveis ao software e ao projeto: padrão de log de erro, de log de informação, padrão de mensagens, metodologia para desenvolvimento do sistema, padrões de projeto (design patterns) a serem aplicados, padrões arquiteturais etc.
Legais	Exigências de conformidade do software com alguma legislação pertinente ao projeto, por exemplo, atendimento a alguma norma da Agência Nacional de Saúde para software de hospital, a norma do Banco Central para sistemas financeiros etc.

Exemplos por categoria

A seguir, vamos ver um exemplo de RNF para cada categoria listada anteriormente.

Muitos profissionais de software acham que um RNF geralmente é algo subjetivo, mas com base nos exemplos a seguir podemos ver que não, basta que sejam especificados com clareza e nível de detalhes suficientes (e necessários) para um bom entendimento. Obs.: para identificar a qual categoria pertence o RNF observe o campo pertinente (campo “Categoria”).

Identificador	RNF01	Categoria	Desempenho
Nome	Tempo limite para processamento de todos os lotes de fatura na rotina diária		
Data de criação	18/01/2016	Autor	Alexandre de Afrodísias
Data da última alteração	N/A	Autor	N/A
Versão	1	Prioridade	Essencial
Descrição	No módulo de faturamento o processamento de faturas em lote é um processo oneroso em termos de memória e CPU		

devido ao alto volume de dados. Em função desta realidade o sistema deverá prover recursos para processamento paralelo (multithreading) que possibilite processar lotes de faturas de forma paralela, compactando o tempo de execução da rotina diária.

A média diária de faturas a serem processadas é 80.000. Cada lote contém 500 (quinhentas) faturas, totalizando 160 (cento e sessenta) lotes. A janela de produção disponível para o processamento de todos os lotes é de 4h.

Considerando as medidas acima, o sistema deve processar todos os 160 lotes em, no máximo, 4h. Para atender isso, o sistema deverá rodar os lotes na quantidade máxima permitida de threads, considerando a seguinte especificação do servidor de aplicativos:

- 16 processadores com quatro núcleos cada.
- 64 GB de memória RAM.
- 1 TB de espaço em disco.

Obs.: deve haver no sistema alguma funcionalidade ou arquivo de configuração onde seja possível o próprio analista da TI parametrizar a quantidade de threads que o sistema deverá rodar. Esta informação não pode ser fixada em código e nem ser de domínio apenas do fornecedor que implementará a solução.

Identificador	RNF02	Categoria	Disponibilidade
Nome	Utilização do módulo de Informações Cadastrais em modo off-line		
Data de criação	25/01/2015	Autor	Aristarco de Samos
Data da última alteração	N/A	Autor	N/A
Versão	1	Prioridade	Importante
Descrição	O módulo de Informações Cadastrais é um módulo do CRM que precisa funcionar 24 x 7 (vinte e quatro horas por dia,		

sete dias por semana) na operação do Call Center da empresa. Por isso é necessário que o sistema possua recursos para sua utilização em modo “off-line”, pois em nossa infraestrutura não é possível ter garantia de 100% de disponibilidade do servidor de banco de dados. Para informação, a garantia atual é de 89% de disponibilidade do ambiente.

Todos os registros de clientes cadastrados no módulo poderão ser mantidos (alterados/consultados/excluídos) com o sistema off-line e novos registros de clientes (inclusão) poderão ser incluídos também com o sistema off-line. Todos os relatórios do módulo de informações cadastrais também precisarão rodar off-line.

Cada usuário do módulo deverá ter em sua estação de trabalho uma cópia do banco de dados do módulo citado, sempre com a mesma versão do modelo de dados utilizado. Deverá haver uma rotina no banco de dados do sistema (banco hospedado no servidor da aplicação) que a cada operação de inclusão/alteração/exclusão de registros nas tabelas do módulo de informações cadastrais sincronize estas atualizações com as bases de dados locais de cada usuário, para manter a massa de dados na mesma posição.

Sempre que o usuário abrir o sistema uma função deverá verificar se há conectividade com o servidor de banco de dados. Se houver, deverá conectar neste ambiente (servidor), senão, deverá conectar na versão do banco de dados local da aplicação.

O sistema deverá ainda ser preparado para fazer sincronização dos dados incluídos/alterados/excluídos quando no uso do banco de dados local (sistema off-line), e na sincronização de “volta” (banco local para banco no servidor), verificar se mais de um usuário manteve um mesmo registro, e realizar merge para que não haja defasagem/perda de dados.

Identificador	RNF03	Categoria	Segurança
Nome	Autenticação de usuário para consumo de webservices do sistema por sistemas externos		
Data de criação	30/01/2016	Autor	André Comte Sponville
Data da última alteração	N/A	Autor	N/A
Versão	1	Prioridade	Essencial
Descrição	<p>Todas as APIs¹¹ do sistema expostas como webservices poderão ser acessadas por sistemas externos de clientes, fornecedores e parceiros. Este acesso precisa ser seguro, com autenticação em nível do servidor e em nível da aplicação.</p> <p>Para autenticação no nível de servidor, o IP de cada consumidor dos webservices deverá ser cadastrado no servidor web onde o sistema estará hospedado, com acesso para execução de scripts. Há uma política de segurança que revisa a validade destes acessos a cada mês, isso deve ser considerado no tratamento de exceções no contexto deste requisito.</p> <p>Para autenticação no nível da aplicação, cada consumidor dos webservices deverá possuir um usuário ativo no sistema. A senha do usuário deverá ser gravada/trafegada utilizando-se o algoritmo SHA-3 para criptografia.</p> <p>O sistema não poderá permitir cache de senha, salvamento de senha ou qualquer outro recurso do tipo. A cada novo acesso, a autenticação deverá ser realizada novamente, de maneira integral.</p> <p>Deverá haver uma política de segurança que assegure que, a cada mês, a senha de cada um dos usuários citados expire e precise ser renovada, e que tenha critérios de complexidade alta de senhas (vide o documento da área de</p>		

¹¹ <http://www.ateomomento.com.br/o-que-e-api/>

	infraestrutura da empresa que tenha detalhes sobre os níveis de complexidade exigidos para cadastro de senhas); tudo isso deve ser considerado no tratamento de exceções no contexto deste requisito.
--	---

Identificador	RNF04	Categoria	Interoperabilidade
Nome	Integração com sistema do Banco Central para envio de informações de transferência internacional de valores		
Data de criação	30/01/2016	Autor	René Descartes
Data da última alteração	N/A	Autor	N/A
Versão	1	Prioridade	Essencial
Descrição	<p>Diariamente, em janela de produção específica, o sistema deverá enviar para o BACEN (Banco Central) as informações do processamento diário (sempre com um dia de atraso, ou D-1) de transferências internacionais de valores realizadas pelo banco.</p> <p>A integração se dará por envio de arquivo texto, a ser confeccionado conforme layout específico para o arquivo CI03. Os dados do layout para o arquivo citado devem ser obtidos em https://www.bcb.gov.br/?INFOL, página do site do BACEN que mantém atualizadas as informações sobre o layout do arquivo CI03. O protocolo a ser utilizado para transmissão é o FTP, mas por razões de segurança do BACEN, a porta utilizada nunca é a 21. A conexão é autenticada, com usuário e senha que o BACEN fornece ao banco. Para o sistema transmissor, o usuário e senha não precisam ser criptografados no ato da abertura da conexão FTP com o BACEN.</p> <p>Deverá haver um recurso no sistema que permita avisar ao administrador do sistema, por e-mail e SMS (sempre ambos), quando a transmissão não for bem-sucedida, detalhando qual a causa do insucesso. Este mesmo recurso deve permitir que, após intervenção humana (seja nos dados que populam</p>		

o arquivo ou nas configurações do sistema), possa ser realizada a retransmissão dos arquivos.

Todo o processo de envio/reenvio do arquivo deverá ser logado. Os logs devem ser gravados em arquivo texto quando o envio for bem-sucedido, e quando não for, em arquivo texto e no Event Viewer do servidor de aplicações onde o sistema estará hospedado.

Para RNFs de integração (categoria Integração) que envolvam layouts de arquivos, estrutura de arquivos xml ou algo do tipo é recomendado que se detalhe a estrutura do arquivo (layout) no corpo do RNF, ou que se faça referência a algum documento que contenha os detalhes de tal estrutura. Quando não há esta especificação é muito comum faltar campo, ter campos com tipos errados, validações não passarem etc. e isso sempre gera muitos problemas.

Identificador	RNF05	Categoria	Usabilidade
Nome	Uso de design responsivo nas interfaces gráficas		
Data de criação	30/01/2016	Autor	Diógenes de Apolônia
Data da última alteração	N/A	Autor	N/A
Versão	1	Prioridade	Importante
Descrição	<p>O sistema de Atendimento a Clientes será construído para rodar em ambiente web. Deverá possuir um design responsivo (https://en.wikipedia.org/wiki/Responsive_web_design).</p> <p>A interface do sistema deverá se comportar adequadamente independente do front-end que será utilizado para acesso – Browser, Smartphone ou Tablet.</p> <p>Obs.: durante o processo de homologação do sistema serão realizados testes de usabilidade validando este requisito. O não atendimento a este requisito gerará o não pagamento relativo</p>		

	à fração pertinente à funcionalidade que não for homologada, segundo os critérios aqui apresentados.
--	--

Identificador	RNF06	Categoria	Compatibilidade
Nome	Compatibilidade com sistemas operacionais Windows e Linux		
Data de criação	30/01/2016	Autor	Friedrich Engels
Data da última alteração	N/A	Autor	N/A
Versão	1	Prioridade	Essencial
Descrição	<p>O parque da empresa é composto, nas estações de trabalho, por sistemas operacionais Linux e Windows. Para Linux a variante utilizada é o Ubuntu a partir da versão 15.10, para Windows utiliza-se versões a partir do XP (XP, Vista 7 e 8), considerando sempre o Service Pack mais recente.</p> <p>Para ambos os sistemas são aplicadas rigorosamente as atualizações dos fabricantes, sempre que são liberadas.</p> <p>O sistema, por se tratar de um aplicativo desktop em arquitetura cliente/servidor, deverá rodar nos sistemas operacionais elencados neste requisito considerando as demais informações aqui descritas. O comportamento deve ser o mesmo para qualquer um dos sistemas operacionais relacionados, tanto no que se refere às funcionalidades quanto à instalação.</p> <p>Obs.: para este requisito haverá garantia do fornecedor do sistema para que os releases do aplicativo mantenham retrocompatibilidade com os sistemas operacionais citados. Esta garantia vigorará enquanto o contrato de manutenção estiver vigente.</p>		

Identificador	RNF07	Categoria	Padrões
Nome	Divisão arquitetural do sistema em camadas para desacoplamento		

Data de criação	30/01/2016	Autor	Empédocles
Data da última alteração	N/A	Autor	N/A
Versão	1	Prioridade	Essencial
Descrição	<p>O projeto do software deverá ser fortemente orientado a baixo acoplamento e alta coesão, primando pela melhor separação de responsabilidades.</p> <p>Todo o projeto deverá ser feito utilizando uma arquitetura separada em camadas, onde cada camada conterá apenas os algoritmos relacionados à sua responsabilidade. Abaixo as camadas que deverão ser utilizadas, e suas responsabilidades:</p> <p>Interface: abrigar lógicas de tela, validação de campos, acionamento de comandos, códigos para design de interface etc. Obs.: Para esta camada deverá ser utilizado o “code behind” de cada tela, não podendo ser criada uma camada “adicional”.</p> <p>Negócio: abrigar lógicas de negócio, onde será codificado o escopo das regras de negócio associadas aos requisitos funcionais pertinentes à funcionalidade.</p> <p>Dados: abrigar lógicas de acesso a dados, comandos SQL ou comandos para utilização de mecanismos de persistência utilizado, para o caso de uso de ORM.</p> <p>Segurança: abrigar lógicas de autenticação, auditoria, manutenção de usuários.</p> <p>Infraestrutura: abrigar lógicas não relacionadas a interfaces gráficas, regras de negócio, dados ou segurança, mas que poderão ser utilizadas em todas estas camadas. Conterá recursos para gravação de logs, transferência de arquivos, mensagens, envio/recepção de e-mails etc.</p> <p>Obs.: em nenhuma das camadas serão permitidos métodos com mais de 40 linhas de código.</p>		

Identificador	RNF08	Categoria	Legais	
Nome	Atendimento à instrução normativa 554 da ANS (Agência Nacional de Saúde)			
Data de criação	30/01/2016	Autor	Hipócrates	
Data da última alteração	N/A	Autor	N/A	
Versão	1	Prioridade	Essencial	
Descrição	Para atendimento à instrução normativa 554 da ANS, o módulo de prontuário deverá gravar em todas as suas tabelas as informações de data/hora do atendimento realizado e dados do médico que realizou o atendimento (CRM e nome completo). Estas informações poderão ser solicitadas pela ANS há qualquer momento, sem aviso prévio.			
	Para atender este requisito, cada tabela do módulo de prontuário deverá conter as colunas abaixo, com as respectivas configurações:			
	Campo	Descrição	Tipo	Obrigatório
	DTHRATEN	Data e hora em que o médico atualizou o prontuário.	Datetime	Sim
	NMMEDICO	Nome do médico que atualizou o prontuário.	Varchar(100)	Sim
	CRMMEDICO	CRM do médico que atualizou o prontuário.	Varchar(10)	Sim
Deverá ser criado um relatório no módulo correspondente do sistema, onde conterá a informação deste requisito. Este relatório será utilizado numa eventual visita dos fiscais da ANS, ou caso a ANS solicite envio de comprovação do atendimento à norma 554.				

Regra de Negócio

Deduzo que antes do lançamento do microcomputador o termo “Regra de Negócio” era algo interpretado totalmente isolado dos softwares empresariais, ou talvez nem fosse um termo conhecido pelas pessoas.

Nos tempos atuais é difícil encontrar alguém que entende Regra de Negócio como algo isolado do software. Quando se fala “Regra de Negócio”, praticamente “sempre” é no contexto de um sistema. É possível uma empresa mais arcaica viver sem software, mas não consegue viver sem regras de negócio.

Para ilustrar isso, imaginemos uma empresa que possui um departamento de expedição de materiais, mas que não possui software para automatizar as atividades deste departamento. Vejamos o cenário abaixo:

Sempre que uma pessoa se dirigir ao departamento de expedição para solicitar uma mercadoria esta pessoa deve se identificar com seu documento de identidade. O profissional do departamento de expedição deve certificar-se que o documento é válido.

Após checar que o documento é válido, o profissional deverá pegar o documento de protocolo de entrega com a pessoa, e neste documento conterà a seção e caixa onde se encontra a mercadoria.

Deverá então dirigir-se à seção, na caixa identificada, pegar o material e levar ao guichê para entrega à pessoa que o solicitou. Antes de realizar a entrega, deverá solicitar que a pessoa assine o livro de entregas, incluindo seu documento e dados de endereço. No livro também devem ser escritos os dados da mercadoria (nome, categoria, marca e modelo), nome do profissional que fez a entrega, e data e hora da entrega.

Se a mercadoria solicitada não estiver na seção e caixa onde deveria estar, o profissional do departamento deverá entrar em contato com a gerência para reportar o problema. O mesmo deve ser feito caso identifique-se que o documento da pessoa que está buscando o material não é válido.

No cenário acima percebemos que a operação do departamento de expedição é viável sem um software, e que existem uma série de **critérios e**

restrições para que o material seja entregue ao seu solicitante. Os critérios e restrições informados são regras, e **regras da empresa** (negócio) que faz as entregas. Logo, são regras de negócio.

Regras de negócio são **restrições** aplicadas a uma operação comercial de uma empresa, que precisam ser atendidas para que o negócio funcione da maneira esperada.

Um software tem como objetivo automatizar atividades de uma empresa. Isso se dará através da criação de funcionalidades, que realizarão requisitos funcionais. Mas os requisitos funcionais, como citado anteriormente, definem **quais são** as necessidades/exigências da empresa em termos funcionais (que funcionarão através de um sistema), ou seja, **o que o sistema deverá fazer**. As regras de negócio definem como o sistema fará o atendimento às necessidades/exigências definidas; uma RN pode ser compreendida quanto a **como um Requisito Funcional se realizará**.

Importância das Regras de Negócio

E raro, muito raro mesmo, encontrar um Requisito Funcional que não possua dependência com uma ou mais regras de negócio. Em função disso, RNs são tão importantes quanto RFs. Um RF não identificado ou não realizado pode gerar um débito técnico¹² sério de escopo, mas uma RN mal especificada pode gerar mais ônus ainda, pois o sistema poderá contrair bugs em função disso.

Ou seja, a funcionalidade existirá, mas estará processando o que tem que processar de maneira errada, e detectar isso após a construção do sistema se a Regra de Negócio estiver especificada incorretamente é algo praticamente impossível, só quando o sistema for para produção e parar na mão do cliente. Isso é o pior cenário possível.

Atributos de uma boa Regra de Negócio

Uma RN com qualidade precisa atender a alguns atributos específicos. Na literatura, tanto nacional quanto estrangeira, não há material (ao menos que eu conheça) que especifique estes atributos para RN. Entretanto, devido à estrutura sintática de uma RN ser muito semelhante à de um RF, eu elenquei alguns atributos (alguns comuns aos RFs) a serem considerados na

12 <http://www.ateomomento.com.br/o-debito-tecnico/>

especificação de uma RN. A seguir a lista dos atributos que considero relevantes.

Atributo	Referente a
Unidade	A RN deve propor/viabilizar uma única coisa apenas. Não deve atender a mais de uma restrição. A RN “Cálculo de salário” não é unitária, pois se refere implicitamente a cálculo de qualquer tipo de salário, e em qualquer empresa existem formas diferentes de calcular salário (para profissional ativo, aposentado, estagiário, efetivo, licenciado etc.). Esta RN assume assim várias responsabilidades, quando deveria assumir apenas uma.
Compleitude	A RN deve ser autocontida, deve ter “início/meio/fim”, ser completa. A RN “Cálculo de salário” não é completa, só conta “parte da história”. Para ser completo deveria ser algo como “Cálculo de salário para profissionais afastados há mais de 15 dias”.
Consistência	Não deve contradizer outra RN do mesmo escopo do projeto. É como termos duas RNs se propondo a fazer uma mesma coisa, mas cada RN se propondo a fazer esta coisa de formas diferentes.
Atomicidade	Uma RN para ser atômico precisa também ter unidade, pois atomicidade remete a assumir apenas uma responsabilidade. Mas também, deve ser indivisível, não podendo ser decomposta. Muitos RNs possuem conjunção, dependem de outras para se realizar. Onde temos duas RNs “Calcular juros para pagamento atrasado” e “Incluir juros para pagamentos de financiamento imobiliário atrasados” na realidade, se pensarmos em atomicidade, temos uma única RN que é “Calcular juros para pagamentos atrasados de financiamento imobiliário”.
Não-Ambiguidade	Não pode ser ambígua, definir algo que não fica claro o que é. A RN “Critérios para processamento de faturas” é ambígua. Fatura de que, critérios para processar o que? “Critérios para processamento de fatura de mensalidade” já é melhor, mas ainda é ruim. Mensalidade de que? Seria não ambíguo se não deixasse dúvidas, algo como “Critérios para processamento de faturas de mensalidades de alunos do segundo grau” ou

Atributo	Referente a
	“Critérios para processamento de faturas de mensalidades de qualquer aluno impendente de série”.
Verificável	Não adianta ter uma RN se ele não é palpável, possível de associar com um RF que será construído, testado etc. Uma RN tem que ser testável, tem que ser possível atestar que a RN foi atendida através de algum RF. Para isso tem que ser também rastreável.
Rastreável	Deve ser possível achar a RN no sistema pronto. Como saber se uma RN foi atendida? Para isso é necessário ter rastreabilidade, e isso só é possível ligando as pontas (associar a RN ao RF, associar o RF à interface gráfica, que será associada a um caso de uso, que será associado a funcionalidades, que serão implementadas etc.).
Exemplificável	Muitas RNs tratam de cálculos, fórmulas, algoritmos etc. Uma RN deve poder ser exemplificada fora do contexto do sistema, para assim facilitar o entendimento de seu escopo pelos profissionais que a implementarão/validarão.

Um detalhe importante é que uma RN não possui prioridade. Como uma RN, no contexto de um sistema, somente existe se associada a um ou mais Requisitos Funcionais, a prioridade aplicada à RN será a prioridade aplicada ao requisito que depende dela.

Estrutura de uma Regra de Negócio

Não há um padrão estabelecido sobre a estrutura de um RN. Mas a maioria das empresas utiliza um formato semelhante, contendo campos específicos. O modelo a seguir contempla os campos mais relevantes, com posterior descrição de cada um.

Identificador	<<Numero>>		
Nome	<<Texto>>		
Data de criação	<<Data>>	Autor	<<Texto>>
Data da última alteração	<<Data>>	Autor	<<Texto>>
Versão	<<Numero>>	Dependências	<<Texto>>

Descrição	<<Texto>>
-----------	-----------

Campo	Descrição
Identificador	Sufixo seguido de um identificador único. O sufixo geralmente utilizado é RN (Regra de Negócio) e o identificador único geralmente é composto de quatro dígitos (podendo ser mais, conforme a o tamanho do sistema que está sendo especificado).
Nome	Nome curto da RN, mas que possibilite entender bem o que RN faz apenas pelo nome.
Data de criação	Data da criação do RN, ou a data em que ela foi especificada.
Autor	Profissional que especificou a RN pela primeira vez, quem a criou.
Data da última alteração	Data em que houve a última alteração no RN.
Autor	Profissional que alterou a especificação da RN pela última vez.
Versão	Número da versão do RN. Geralmente utiliza-se algo simples, como 1, 2 etc. A versão inicial sempre é a 1, e a cada alteração incrementa-se a versão (na criação versão 1, na primeira alteração versão 2 etc.).
Dependências	Quais RFs são dependentes da RN para serem realizados. Coloca-se apenas o identificador dos RFs.
Descrição	Descrição detalhada (a mais detalhada possível) da RN.

Exemplos

Vejamos alguns exemplos de RN. Para os exemplos utilizei o cenário descrito anteriormente para a empresa que possui o departamento de expedição de materiais (entregas). Os RFs que dependem destas RNs não estão especificados, são fictícios. Utilizaremos apenas o identificador dos RFs (RF0099 – Realizar entrega presencial de material no departamento de expedição e RF0002 – Informar por e-mail o gerente do departamento de expedição sobre ausência de material no almoxarifado).

Identificador	RN0001		
Nome	Validação da identificação da pessoa que solicita a retirada/entrega do material		
Data de criação	31/01/2016	Autor	Nagarjuna
Data da última alteração	N/A	Autor	N/A
Versão	1	Dependência	RF0099
Descrição	<p>Sempre que uma pessoa se dirigir ao departamento de expedição para solicitar uma mercadoria esta pessoa deve se identificar com seu documento de identidade. O profissional do departamento de expedição deve certificar-se que o documento é válido.</p> <p>Para validar o documento fornecido pela pessoa o número do documento deverá ser validado no sistema da Secretaria de Segurança Pública do Estado de São Paulo, através de funcionalidade correspondente no módulo de controle de expedição. Se o documento não tiver como órgão emissor SSP-SP, não precisará ser validado, mas deverá ser microfilmado e ter uma cópia armazenada no sistema, através de funcionalidade específica.</p>		

Identificador	RN0002		
Nome	Localização automática do material solicitado para ser entregue		
Data de criação	31/01/2016	Autor	Nicolau de Cusa
Data da última alteração	N/A	Autor	N/A
Versão	1	Dependência	RF0099
Descrição	<p>Após checar que o documento é válido (vide RN0001) o profissional deverá pegar o documento de protocolo de entrega com a pessoa.</p> <p>O número do protocolo entregue pela pessoa que solicitou o material deverá ser informado no sistema, que após inserção</p>		

	do número do protocolo deverá informar qual a seção e caixa em que o material se encontra.
	Se a pesquisa através do identificador do protocolo não informar a seção e caixa em que o material está armazenado, o sistema automaticamente deverá enviar um e-mail ao gerente do departamento para ciência e providências, informando qual é o material (número de registro, marca e modelo), quem depositou o material na seção/caixa quando o material chegou ao almoxarifado, em qual seção/caixa o material deveria estar, data e hora em que o protocolo foi emitido, e data e hora em que a entrega foi solicitada.

Identificador	RN0003		
Nome	Formalização da entrega do material solicitado		
Data de criação	31/01/2016	Autor	Parmênides
Data da última alteração	N/A	Autor	N/A
Versão	1	Dependência	RF0099, RF0002
Descrição	<p>Após a coleta do material a ser entregue pelo profissional do departamento de expedição, a entrega deverá ser formalizada via sistema, onde ocorrerá o registro da liberação do material através de um termo de liberação de material.</p> <p>No termo de liberação de material a ser emitido, deverá conter as seguintes informações: nome e e-mail da pessoa à qual o material foi entregue, nome do profissional que entregou o material, nome, categoria, marca e modelo do material, data e hora em que a entrega foi realizada.</p> <p>O termo de liberação deve ser impresso e assinado pela pessoa que recebeu o material, e posteriormente armazenado em local apropriado (não deverá ser armazenado no sistema).</p>		

Diferenças entre Requisito Funcional e Regra de Negócio

Eu imagino que antes do lançamento do microcomputador, o termo “Regra de Negócio” era algo interpretado totalmente isolado dos softwares empresariais, ou talvez nem fosse um termo conhecido pelas pessoas.

Nos tempos atuais é difícil encontrar alguém que entende Regra de Negócio como algo isolado do software. Quando se fala “Regra de Negócio”, praticamente **sempre é** no contexto de um sistema.

Estes dois conceitos (Requisito Funcional e Regra de Negócio) se encontram (se cruzam) a toda hora na modelagem de um sistema, mas são coisas diferentes. É possível uma empresa mais arcaica viver sem software, mas mesmo uma empresa arcaica **não consegue viver sem regras de negócio**.

As regras de negócio são **restrições/premissas** necessárias para o negócio “acontecer”. E estão em todo lugar, inclusive no nosso corpo. Neste post, mais à frente, vamos ver dois exemplos superficiais de regras de negócio contidas em nosso corpo, e de requisitos funcionais relacionados.

Nosso corpo não é bem uma empresa, mas pode ser encarado desta forma. Somos o responsável por administrá-lo, e se cuidarmos dele como um bom empresário cuida de seu negócio, provavelmente teremos excelentes resultados.

Por agora, vamos pensar num negócio mais **palpável**. Vamos pensar num negócio de venda de coco na praia, executado e administrado por um vendedor, que é uma pessoa física.

Este vendedor pode controlar suas vendas num aplicativo de seu smartphone; sim, pode.

Mas pode também manter seu negócio sem isso (sem automatizar seu controle através de um aplicativo num smartphone). **Mas ele não conseguirá trabalhar sem ter suas regras de negócio**, mesmo que ele não saiba o que é uma Regra de Negócio.

Exemplo num negócio real



Vejam os abaixo algumas regras de negócio inerentes ao negócio de “vender coco na praia”:

- O coco somente será entregue ao cliente que realizar o pagamento.
- Somente serão aceitos pagamentos em dinheiro vivo. Não serão aceitos cartões de crédito, cartão de débito ou cheque.
- Clientes que comprarem 4 cocos ganharão um coco de graça. Esta promoção não terá data para término.
- Quando a caixa de isopor (onde ficam os cocos) ficar sem gelo o vendedor deverá parar, abastecer a caixa com gelo e somente aí continuar a vender.
- O vendedor deverá portar ferramenta que permita furar o buraco no coco para que o cliente possa bebê-lo. Esta ferramenta não pode ser cortante nem serrilhada nas bordas.
- O vendedor deverá portar canudinho e fornecê-lo ao cliente ao entregar-lhe o coco, para que o cliente possa bebê-lo. Sempre deverá oferecer o canudinho, mas deverá abri-lo após o cliente aceitar recebe-lo, pois se o cliente não quiser, uma vez não aberto o canudinho (não tirado o plástico), não haverá desperdício deste material.

Poderíamos elencar dezenas de outras regras de negócio do contexto apresentado, mas com as descritas já fica claro que **Regra de Negócio existe sem sistema**, e que uma empresa **não existe** sem regras de negócio.

Sistemas não existem sem regras de negócio e nem todas as regras de negócio são automatizadas através de um sistema. Pode até ter um sistema sem regras de negócio, apenas com requisitos funcionais por exemplo, mas seria um sistema que permitiria fazer muita coisa de qualquer jeito, e até hoje acho que ninguém fez isso pois seria algo muito caótico. E nem toda Regra de Negócio é realizada por um sistema. Existem procedimentos (por exemplo – pegar café no escritório sempre que ficar com sono após o almoço) que geralmente não são automatizados via software.

Se o vendedor de coco contratar um profissional para implementar um software para ajudá-lo nas vendas, parte destas regras deverão ser atendidas no sistema. **Mas não serão requisitos funcionais, serão regras de negócio.**

Diferença de Requisito Funcional e Regra de Negócio

A diferença entre Requisito Funcional e Regra de Negócio, conceitualmente falando, é que o Requisito Funcional se refere à **o que** o sistema deverá fazer, enquanto a Regra de Negócio refere-se a **como** o sistema deverá fazer.

Do ponto de vista do negócio (negócio do cliente para o qual o sistema está sendo feito), ambos são necessidades (Requisito Funcional e Regra de Negócio), mas cada uma com um **foco diferente**.

Uma boa dica para saber o que é Regra de Negócio é perceber quando há condições: “somente, quando, requer, se, obrigatório, sempre”. Requisitos não possuem condições, regras “são” condições. Requisitos são ações objetivas, desejo, solicitação.

Mais sobre a Venda de Coco

Ainda no exemplo do vendedor de coco, teríamos requisitos funcionais como:

- Processar venda de coco.
- Aplicar promoções vigentes na realização da venda.
- Calcular quantidade de gelo conforme o tamanho da caixa de isopor.
- Calcular quantidade de canudinhos para a quantidade de cocos contidas na caixa de isopor.
- Incluir desconto padrão na venda de cocos.

Vemos acima que os requisitos funcionais são de fato “o que” o sistema deverá fazer, tratando das necessidades, desejos, solicitações a serem materializadas no sistema. Como isso será feito, no que diz respeito às restrições de negócio, as regras dirão.

No dia a dia, o Analista de Sistemas tende a confundir as duas coisas, e isso gera uma séria de prejuízos ao projeto. Dois bem sérios são:

- Impossibilidade de se fazer **reuso** de Regras de Negócio: vários requisitos frequentemente realizam regras de negócio comuns. As regras de negócio são associadas a requisitos, que as realizam. Não havendo reuso, é fatal que haverá mais de uma Regra de Negócio com o mesmo escopo (**regras repetidas**). E provavelmente, isso será codificado **mais de uma vez** no software, uma vez que o norte do desenvolvimento são os requisitos.

No escopo/projeto de um sistema, nada deve ser repetido, isso deve ser mantra para um profissional da área. Essa boa prática/premissa é uma das mais importantes, pois se uma mesma coisa está em dois lugares, uma mesma coisa foi feita duas vezes e gerou custo dobrado para isso; são dois pontos de bug em potencial, são dois pontos para manutenção de uma mesma coisa etc.

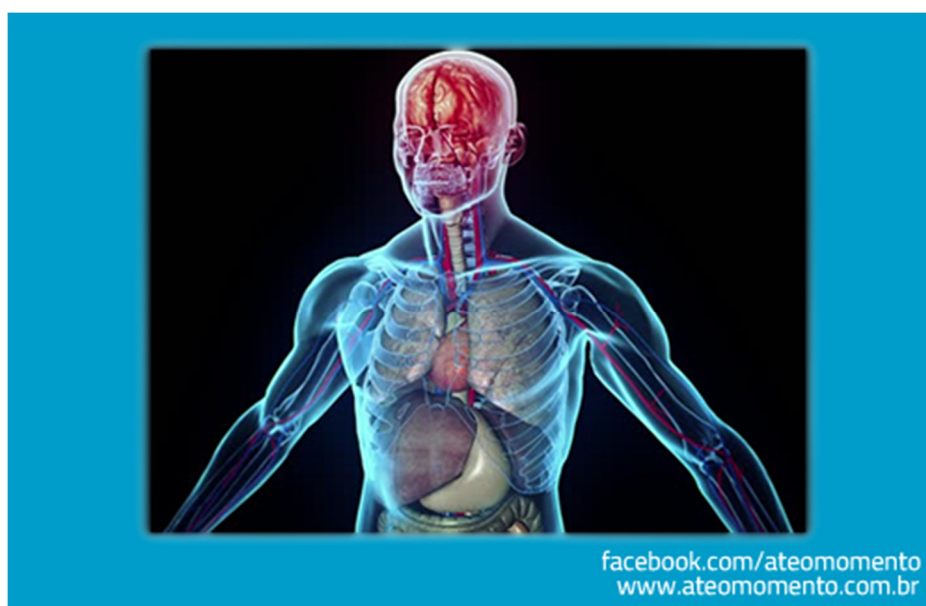
– Violação do **princípio da responsabilidade única**¹³ (princípio que se aplica a qualquer artefato de produção de um software, não somente a modelos de código fonte): Uma Regra de Negócio tem a responsabilidade de **restringir** algo, baseado na condição que é considerada em seu escopo. Um Requisito Funcional é um **objetivo** que o sistema deverá alcançar, uma função que o sistema deverá realizar.

Misturar isso faz com que um requisito assuma a responsabilidade de uma regra, e vice-versa. Essa “mistura” gera **alto acoplamento** em nível de especificação (pois se não se separa um do outro, naturalmente que o escopo ficará misturado também), o que dificulta separar as responsabilidades dos requisitos e regras adequadamente, favorecendo assim que na construção seja gerado um sistema com **forte acoplamento e baixa coesão** (pois a especificação é insumo para construção).

Dezenas de problemas poderiam ser descritos, mas os citados acima (acho) já dão uma ideia do prejuízo causado por esta confusão.

No final, são conceitos simples (Requisito Funcional = **o que o sistema fará** – Regra de Negócio = **como o sistema fará**), que se bem entendidos favorecem muito o sucesso do projeto. Pode parecer um pouco complicado no início, mas a partir da prática vai ficando mais claro, menos abstrato.

Exemplos no Corpo Humano



¹³ <http://www.ateomomento.com.br/s-o-l-i-d-srp-single-responsibility-principle/>

Nosso corpo é uma máquina perfeita. Através da observação¹⁴ podemos aprender tudo analisando o corpo, de análise de sistemas à artes.

Temos o sistema digestivo. É comum associarmos isso ao estômago, principalmente. Nosso estômago tem como função principal digerir os alimentos.

E sabemos que, quando nosso estômago está cheio (cheio mesmo, no limite), se enviarmos mais alimento para digestão, este alimento não conseguirá ficar armazenado, então, deverá ser colocado para fora (isso acontece pelo famoso vômito).

Temos então, a grosso modo, um Requisito Funcional e uma Regra de Negócio.

Requisito Funcional

Digerir os alimentos inseridos através da boca e transportados pelo tubo digestivo.

Regra de Negócio

Expelir alimentos pelo tubo digestivo quando houver o preenchimento de 100 por cento do espaço do estômago.

Consideremos ainda outro contexto presente no corpo humano, ainda no mundo do “comer”. Quando engolimos o alimento de maneira “errada”, ou tentamos engolir algo que não passa pela traqueia, ocorre o engasgo. Vejamos um Requisito Funcional e uma Regra de Negócio neste contexto.

Requisito Funcional

Absorver ar pelas vias aéreas a partir de inalação pelo nariz.

Regra de Negócio

Viabilizar um engasgo quando houver bloqueio das vias aéreas por entupimento.

¹⁴ <https://pt.wikipedia.org/wiki/Observação>

Um exemplo no software, para fechar

Vejamos o programa **Ping**¹⁵, existente em qualquer sistema operacional. Este programa serve para verificar se há conectividade entre o host “local” e um host qualquer. Essa conectividade é verificada através do envio de pacotes ICMP¹⁶ para o host destino, e se este host receber o pacote envia um retorno que o programa entende e interpreta.

Supondo que estamos especificando o programa Ping – logo este é o nosso escopo – podemos então identificar (com base nas imagens a seguir – o programa faz mais coisas além do que descrevi) os seguintes Requisitos Funcionais:

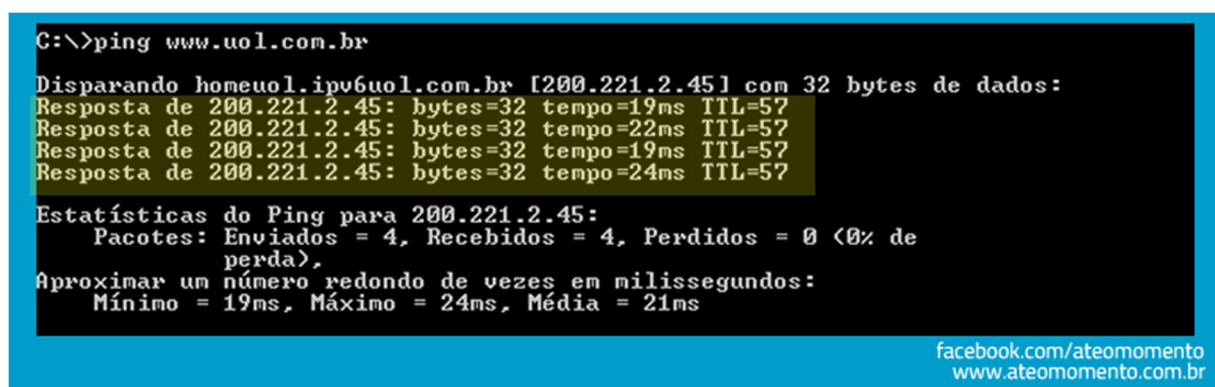
Requisito Funcional

Enviar pacotes ICMP para o host destino e monitorar o retorno do envio.

Requisito Funcional

Exibir estatísticas do envio dos pacotes ICMP ao término da execução do programa.

E, para cada contexto representado pelas imagens, algumas regras de negócio também.



```
C:\>ping www.uol.com.br

Disparando homeuol.ipv6uol.com.br [200.221.2.45] com 32 bytes de dados:
Resposta de 200.221.2.45: bytes=32 tempo=19ms TTL=57
Resposta de 200.221.2.45: bytes=32 tempo=22ms TTL=57
Resposta de 200.221.2.45: bytes=32 tempo=19ms TTL=57
Resposta de 200.221.2.45: bytes=32 tempo=24ms TTL=57

Estatísticas do Ping para 200.221.2.45:
    Pacotes: Enviados = 4, Recebidos = 4, Perdidos = 0 (0% de perda),
Aproximar um número redondo de vezes em milissegundos:
    Mínimo = 19ms, Máximo = 24ms, Média = 21ms
```

facebook.com/ateomomento
www.ateomomento.com.br

Nesta imagem, temos um retrato da execução de um Ping do meu host para o host do meu blog. O teste não foi bem-sucedido, pois os pacotes ICMP expiraram (TTL¹⁷ estourou).

¹⁵ <https://pt.wikipedia.org/wiki/Ping>

¹⁶ https://pt.wikipedia.org/wiki/Internet_Control_Message_Protocol

¹⁷ <https://www.vivaolinux.com.br/dica/Entendendo-o-campo-TTL-do-Ping>

Regra de Negócio

O envio dos pacotes ICMP deverá se repetido quatro vezes por cada execução do programa. Mesmo que o TTL dos pacotes ICMP do primeiro envie esgote, as três tentativas de envio restantes deverão ser executadas.

```
C:\>ping www.naoexisteodominio.com.br
A solicitação ping não pôde encontrar o host www.naoexisteodominio.com.br. Verif
ique o nome e tente novamente.
```

facebook.com/ateomomento
www.ateomomento.com.br

Nesta imagem, não foi possível sequer realizar a resolução do nome (traduzir o nome do domínio em IP [DNS¹⁸]).

Regra de Negócio

Interromper execução quando não for possível realizar a tradução do nome do domínio para um endereço IP.

```
C:\>ping www.uol.com.br

Disparando homeuol.ipv6uol.com.br [200.221.2.45] com 32 bytes de dados:
Resposta de 200.221.2.45: bytes=32 tempo=19ms TTL=57
Resposta de 200.221.2.45: bytes=32 tempo=22ms TTL=57
Resposta de 200.221.2.45: bytes=32 tempo=19ms TTL=57
Resposta de 200.221.2.45: bytes=32 tempo=24ms TTL=57

Estatísticas do Ping para 200.221.2.45:
    Pacotes: Enviados = 4, Recebidos = 4, Perdidos = 0 (0% de
    perda),
    Aproximar um número redondo de vezes em milissegundos:
    Mínimo = 19ms, Máximo = 24ms, Média = 21ms
```

facebook.com/ateomomento
www.ateomomento.com.br

E nesta imagem, a execução foi bem-sucedida, o programa conseguiu realizar a tradução do nome para o domínio, enviou os pacotes ICMP, e obteve o retorno no tempo limite.

Regra de Negócio

Interromper a execução quando o tempo de vida (TTL) dos pacotes ICMP enviados expirar.

¹⁸ https://pt.wikipedia.org/wiki/Domain_Name_System

Concluindo

Vimos que os dois conceitos citados são coisas diferentes, e não sutilmente bem diferentes, são muito diferentes.

Priorização de Requisitos

Quando falamos em **Escopo do Sistema** (não **Escopo do Projeto**), estamos falando de requisitos (tanto funcionais quanto Não Funcionais, e também regras de negócio)

A grosso modo, vamos entender como requisito tudo aquilo que deve ser feito no sistema, que compõe o escopo do sistema (o que já vimos anteriormente neste eBook).

Escopo do projeto é algo diferente de escopo do sistema, em projetos de desenvolvimento de software. No escopo do projeto geralmente temos mais coisas além do sistema em si – atividades de infraestrutura, gestão, configuração etc., por exemplo. No escopo do sistema, “que é parte” do escopo do projeto, como citado no parágrafo inicial deste post, o que temos são requisitos, basicamente. Na literatura de gestão de projeto, o escopo do sistema é chamado “Escopo do Produto”.

O Cone da Incerteza¹⁹ nos explica isso melhor, mas outros fatores também contribuem para a problemática do escopo; abaixo alguns para exemplo.

Requisitos mal especificados

No início do projeto geralmente os requisitos tem um nível de detalhe muito alto, **muito macro**. A partir do início do projeto, à medida que os requisitos vão sendo detalhados, sendo decompostos, o escopo “de verdade” vai aparecendo, e na **maioria das vezes** não há tempo de replanejar/recombinar as coisas quando o tempo já passou.

Informações faltantes

Durante a especificação a equipe faz reunião com os usuários, obtêm leis, normas, e-mails etc. Após a conclusão da especificação, ocorre a **validação**, e após algumas idas e vindas, o escopo é validado. Na hora da “materialização” dos requisitos (leia-se projeto, construção, testes etc.) começa

¹⁹ <http://www.ateomomento.com.br/o-problema-da-descoberta-do-escopo-o-cone-da-incerteza/>

o “puxa, esqueci, tinha isso também” e o “puxa, isso aqui é um pouco diferente do que tínhamos pensado”.

“Dinamismo” dos usuários

Usuários **mudam de ideia** muito rápido²⁰, e gestores geralmente tem dificuldades em **dizer não** a usuários.

Definir conceitualmente um sistema é um processo de **criação constante**; ocorrem definições, revisões e ajustes de escopo há todo momento. Mas dificilmente isso é **previamente alinhado** com o cliente, não há um trabalho de aculturação do cliente neste sentido (no sentido de que, se mudar toda hora, nunca fica pronto).

Em função dessa característica, da volatilidade do escopo, usuários tendem a pedir mudanças frequentemente, com o projeto em execução, geralmente não percebendo o impacto que isso gera.

Gestão e controle inadequados do escopo

Controlar o escopo de um projeto é algo muito difícil, pela própria complexidade inerente. Soma-se a isso o fato de que gerentes de projeto e analistas sempre estão envolvidos em vários projetos/demandas ao mesmo tempo – o que dificulta muito focar numa única coisa, e pressões externas e internas que sempre existem em qualquer empresa/mercado dificultam ainda mais o foco.

E nesse ambiente desafiador, a **ausência de critérios** na gestão do escopo do sistema tornam as coisas ainda mais complicadas. Uma forma de diminuir este problema, dando um pouco mais de ordem ao caos, é utilizar a **Priorização de Requisitos**.

O que é mais importante vem primeiro

Todos nós deveríamos refletir muito sobre a palavra **priorização**. Considerando que na vida os recursos mais necessários são **escassos**, é fundamental priorizarmos as coisas. Exemplos facilmente perceptíveis de escassez de recursos são dinheiro, tempo e saúde. E na vida, geralmente fazemos aquilo que queremos, e não aquilo que devemos fazer.

²⁰ www.ateomomento.com.br/relacao-com-o-usuario/

Se analisássemos o que realmente é importante/urgente, com certeza viveríamos melhor, pois faríamos primeiro aquilo que **realmente** é importante. Mas se não houvesse limitação/escassez de recurso, **todos iriam querer tudo!**

Para muitos profissionais envolvidos em projetos de software – tanto fornecedores quanto clientes – quando o assunto é escopo, não há limitação/escassez de recurso, **eles querem tudo!**

Mas não dá para fazer tudo em um projeto. E para decidir o que deve ser feito com os recursos que se tem, uma das melhores formas para isso é **priorizar** os requisitos. Em termos de método aplicado, a “grosso modo”, estamos falando de criar uma lista com os requisitos, definir uma prioridade para cada um, e os mais prioritários vão para o início da lista, os menos prioritários, para o fim. Após a priorização, é importante definir uma ordem de execução para os requisitos com a mesma prioridade.

Priorização dos Requisitos

As priorizações podem ter nomes diferentes conforme a literatura consultada, mas geralmente quando falamos de priorização de requisitos, são utilizados os termos **Essencial**, **Importante** e **Desejável**. A seguir, o que significa cada um destes.

Essencial

Realmente é fundamental para o sistema, sem o qual o sistema não pode ser dado como “completo”, ou “apto para produção”. São requisitos que se não são implementados **impedem uma implantação** ou a conclusão do sistema. São **compulsórios**, não sendo possível aplicar soluções de contorno ou paliativos para eles.

Importante

Deve ser parte do escopo, mas **não bloqueia o sistema a entrar em produção**. É como se o sistema ficasse com uma “pendência” de escopo – criando débito técnico²¹ – que será atendido em momento oportuno. Sem um requisito importante, o sistema **poderá rodar, funcionar, ser**

²¹ <http://www.ateomomento.com.br/o-debito-tecnico/>

utilizado. Pode ser simplesmente postergado para pós-implantação, ou ser atendido temporariamente por soluções de contorno ou paliativos.

Desejável

Não é indispensável para o sistema estar completo, para entrar em produção. Também não é algo que, mesmo postergado, deverá ser feito obrigatoriamente.

Sem um requisito desejável o sistema **deve funcionar de maneira satisfatória**, atendendo completamente seu objetivo. Por ser algo que não precisa ser feito para que o sistema esteja completo, é a menor das prioridades, e deve ser postergado para, se possível ser viabilizado no futuro.

Quando priorizar os Requisitos?

A priorização dos requisitos deve ocorrer, no mínimo, **durante a definição do escopo** do sistema. Isso é o mínimo mesmo. Começar o desenvolvimento com o escopo definido, mas não priorizado, torna **fatal** a ocorrência de problemas diversos no projeto.

Mas priorizar os requisitos **apenas** no início do projeto não é boa prática.

Sabemos que em projetos de software o escopo é volátil, e muito dessa volatilidade origina-se nos desejos/percepção de valor dos usuários, sentimentos que são descobertos/mudam a todo momento. Em função disso, a prioridade dos requisitos também pode mudar. Por isso a necessidade de **priorização constante**, ou **repriorização**.

Em projetos baseados em ciclo de vida em cascata (waterfall), colocar **marcos** no planejamento para revisão da priorização dos requisitos é uma forma útil para manter o escopo mais aderente à realidade do negócio. Excetuam-se os requisitos que já foram implementados, ou seja, a “repriorização” ocorrerá **somente nos requisitos ainda não implementados** (requisitos ainda contidos no backlog do projeto).

Salvo raras exceções, não é recomendada a utilização de ciclo de vida em cascata em projetos de software, a abordagem deve ser iterativa e incremental. Em projetos onde o ciclo de vida se baseia neste modelo (iterativo e incremental) a priorização também deve **ocorrer no início do projeto** (no

backlog do produto/lista de requisitos) e também **sempre** no início de cada iteração, para “repriorização”.

Por iteração entenda-se fase, pacote, Sprint ou qualquer outro rótulo empregado a uma iteração de projeto

Em projetos onde o ciclo de vida utiliza Scrum, por exemplo, é muito útil nas reuniões de Sprint Planning haver **revisão do backlog** – onde são avaliados/reavaliados tanto os itens do backlog, quanto o esforço para produção e a priorização definida para cada item.

De um modo geral, é fundamental revisar as prioridades aplicadas aos requisitos. As coisas mudam muito rápido.