Reusabilidade e portabilidade

A importância da reusabilidade e portabilidade de código existente

Saiba técnicas de reusabilidade e portabilidade que tornam os projetos mais amadurecidos e com visão de reaproveitamento de idéias

Caio Costa, George Dias, Lauriza Santos, Barbara Aniele, Ayran Costa e Daniele Guimarães

Desde o começo da década de 80, a Engenharia de Software tem se tornado uma disciplina amadurecida nas universidades e no mercado. Porém, ao estágio que chegou atualmente, é impreciso afirmar que existem métodos consolidados que podem ser usados em desenvolvimento de qualquer software. Pois a heterogeneidade das necessidades do usuário aumenta a complexidade de uma documentação que estabeleça padrões, mas não quer dizer que não há similiaridade nos projetos e nos requisitos dos usuários, como os projetos tipo CRUD (Criar, Ler, Editar e Remover) que são um grande exemplo dessa situação.

Com o começo da década de 90 viu-se a existência de um conjunto de soluções que conseguia solucionar problemas que eram recorrentes em vários projetos. Estas soluções iam além de código: linguagem de representação do projeto, relacionamento entre entidades, boas práticas de código, quais métodos deveriam ser usados no desenvolvimento.

No que tangia a código e a elaboração de projetos apareceram questões pertinentes: Se um mesmo problema aparece em outro projeto, como pode ser reaproveitada a solução do primeiro projeto sem necessitar fazer tantas mudanças? Será que já existe solução para o meu problema em outros projetos? É possível aproveitar desenvolvedores novatos no projeto sem precisar oferecer um treinamento para passar um conjunto de técnicas que são aprendidas por anos e anos de experiência? Como é possível fazer com que um software não envelheça tão rápido? Há uma possibilidade de que a manutenção que não precise temer tantos efeitos colaterais? Há possibilidade de ser construído um software não só funcional, mas de qualidade? Como é possível estar preparado para as mudanças de requisitos?

O foco na reusabilidade e portabilidade das soluções traz possibilidades de benefícios que contribuem para o desenvolvimento não ser custoso e baseado no empirismo. Pois antes da década de 80 o desenvolvimento não se importava com a "vida útil"(principalmente com o descarte muito cedo), devido ao problema de adição de novas funcionalidades a este. Com a chegada de conceitos focados na flexibilidade (Refatoração, Padrões de Projeto, Padrões Arquiteturais, paradigmas de linguagem focados no reuso), o desenvolvimento de software deixou de ser um exemplo de negócio tendencioso ao fracasso. Assim a discussão de reutilização de projetos se tornou obrigação das empresas de software.

Objetivo

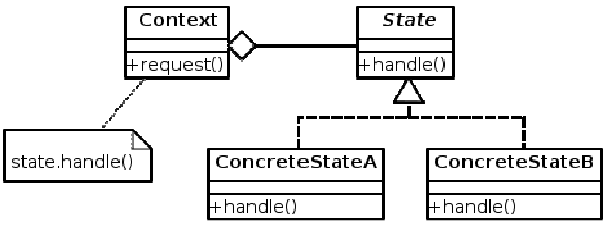
Neste artigo serão apresentadas as abordagem do desenvolvimento de software quanto a reutilização de código existente, apresentando técnicas de reutilização de código e um breve descritivo delas.

Fundamentos

Quanto a portabilidade

Segundo (BRAUDE,2005), devemos projetar pensando em mudanças, pois raramente o requisitos iniciais serão constantes até o final do projeto. Por isso deve-se construir software com possibilidades de flexibilidade, "entretanto, não seria razoável esperar que o projeto monitore atividades não relacionadas:'flexibilidade' tem limites"(BRAUDE,2005).

Ainda (BRAUDE,2005) sugere, como exemplo, que prepare o software com um conjunto de interfaces que estabeleçam comportamentos comuns e que essas interfaces sejam agregadas por algum componente que as utilizem. A flexibilidade é aumentada, pois o conjunto de soluções concretas que implemente essa classe agregada será abstraído, e somente se, tiverem conhecimentos destas no momento de execução da aplicação. Utilização sugerida por (GAMMA,2000) com o padrão de projeto *State*("permite a um objeto alterar seu comportamento quando o seu estado interno muda. O objeto parecerá ter mudado sua classe"). Essa solução é uma forma de diminuir os problemas com herança de implementação (baseada "extends" de Classes): alta redundância, baixa coesão, forte acoplamento e explosão de classes(SHALLOWAY,2004).



Estrutura do Pattern State(http://en.wikipedia.org/wiki/File:State_Design_Pattern_UML_Class_Diagram.svg)

O (KERIEVSKY,2008) afirma que um pensamento de flexibilidade deve sempre focar em um "código menos irritante", mas ele ressalta que deve se tomar cuidado com a "ansiedade" de utilizar padrões em demasia e a possibilidade de se perder muito tempo no projeto. Ele sugere que uma boa prática seria a utilização de Refatoração, pois pode se encontrar softwares que necessitem ser flexíveis e portáveis mas por alguma razão, seu redesenvolvimento é caro (ex: software especialista) ou eles são críticos (A manutenção deve ser dada em runtime).

Quanto a reusabilidade

Cada vez mais empresas de software vêem seu software como um ativo valioso e estão promovendo o reuso para aumentar seu retorno sobre investimentos(SOMMERVILLE,2007). Por muitos anos, a reutilização foi algo que não poderia ficar fora do pensamento de algum profissional: mas só muito recentemente os engenheiros de software aprenderam a alcançar a fazer a reutilização em escala significativa, como por exemplo a utilização disseminada das APIs Java(BRAUDE,2005).

O (BOOCH, 1998) recomenda-se que sempre pense no desenvolvimento de um software a pergunta: "Esse comportamento já foi usado em mais algum contexto?".

As primeiras instâncias da abordagem da reutilização vieram com a documentação e publicação de algoritmos fundamentais(KNUTH, 1971) e, mais tarde com a documentação de tipos de dados abstratos como pilhas, árvores listas (BOOCH, 1998) e mais recentemente com a evolução da orientação a objeto perceberam que determinadas soluções de projeto poderiam ser aplicadas em diferentes casos. Assim como forma de padronização implantou-se a utilização de padrões de projetos (design patterns).

Conteúdo

A reusabilidade apesar de ser um recurso de estratégia de desenvolvimento, ainda não muito é amadurecida, deveria ser discutida na Engenharia de Software como algo natural que ocorresse com frequência em um projeto. Ela vem em prol da agilidade, manutenção do software e redução de custos de produção. Entre algumas vantagens, podem ser citadas:

Confiança aumentada - Software reusado, experimentado e já testado em sistemas de trabalho deve ser mais confiável do que software novo, pois seus defeitos de projetos e implementação já foram encontrados e corrigidos;

Conformidade com padrões - alguns padrões podem ser implementados como um conjunto de componentes reusáveis padronizados;

Uso eficiente de especialistas - Ao invés de desenvolver o mesmo trabalho repetidas vezes, esses especialistas podem depositar seus conhecimentos em softwares reusáveis.

Dentre as desvantagens estão:

Custo de manutenção aumentados - Se o código-fonte de um sistema ou componente não estiver disponível, então o custo de manutenção poderá ser aumentado.

Síndrome de não inventado aqui - Alguns desenvolvedores preferem desenvolver por si mesmo ao invés de reutilizar um software.

Aumentar a reusabilidade do software é considerada como pré condição técnica crucial para melhorar a qualidade geral do software e reduzir os custos de manutenção e de produção.

Design Pattners

Reusabilidade é uma consequência de um conjunto de práticas para que o código seja flexível. Quando se trabalha com Padrões de Projeto, se pensa na melhor forma de deixar o código reutilizável em várias situações, mas o que mais chama atenção é como é tratado a Ligação Dinâmica(Dynamic Binding). Nela envolve o atendimento de uma solitação em tempo de execução, deixando poucas coisas em hard code. Através da ligação dinâmica é possível fazer manutenção do código sem compilar nada e expandi-lo a níveis extremos com Reflexão Computacional.
Uma outra questão muito comum em orientação a objetos é a responsabilidade ou interesse(concern). Quando se pretende implementar um sistema flexível, de fácil acréscimos de

mudanças e que aproveite o máximo da POO, o projeto de responsabilidades de objetos é pensado com cuidado devido a uma questão bem simples ou não: “O que poderá mudar amanhã?”
A reusabilidade acontece quando existe algo já pronto, mas não impede-se que planeje como os módulos serão implementados futuramente focando em uma divisão de responsabilidades clara. Para se aproveitar o máximo da POO, sem especificar alguma linguagem, (GAMMA, 2000) sugeriu os grupos responsabilidade como:

- Criacionais (Criar objetos em configurações variáveis determinadas em tempo de execução);
- Estruturais (Representar árvores variáveis de objetos ou outras estruturas em tempo de execução);
- Comportamentais (Mudar, recombinar ou, de outro modo, captar o comportamento mútuo de um conjunto de objetos).

Sendo 23 padrões de projeto se dividindo nessas três categorias.Ou como (METSKER, 2008)classificou os padrões focando a estrutura: de interface, responsabilidade, construção, operação e extensão.

Dentre todos os padrões, o que chama bastante atenção é o Memento(GAMMA, 2000), pois ele consegue acrescentar metódos as classes.Assim, tais metódos podem ser escritos a qualquer hora e serem adicionados na Classe(uma pseudo reflexão computacional).

Os Padrões de Projeto oferecem uma forma de construção de relacionamento entre as classes que faz surgir o conceito de Framework.

Frameworks de aplicação

A estrutura de trabalho usada para aperfeiçoar a produção e suprir a demanda crescente do software, é uma técnica aplicada do projeto ao desenvolvimento, da reutilização da linha de código a analise de projeto, ou seja, é passível de reutilização e adaptação. (BROOCH, 2000) define como, “Um padrão arquitetural que provê um ‘template’ extensível para aplicações dentro de um certo domínio.“ E de forma mais ampla (QUINAIÁ,2005):

Framework: é um pacote projetado para ser importado com substituições. Ele desdobra-se para prover uma versão de seu conteúdo que é especializado através das substituições feitas. Um framework pode instanciar a descrição de um Padrão, uma família de Padrões mutuamente dependentes, uma colaboração, um refinamento de Padrão, a própria construção do modelo e mesmo um conjunto de propriedades genéricas fundamentais. Frameworks podem ser reconstruídos em outros frameworks.

Características

Uma característica importante dos frameworks é que os métodos definidos pelo usuário para especializá-lo são chamados de dentro do próprio framework, ao invés de serem chamados do código de aplicação do usuário.

- Modularidade
- Reuso
- Extensibilidade

Classificação do framework:

· System infrastruture frameworks (Os frameworks de infra-estrutura)
· Middleware integration frameworks (Os frameworks de integração de “middleware”)
· Enterprise application frameworks (Os frameworks de aplicação empresarial)

Experiência Prática

Foi feita entrevista com Tiago Cajaíba, Arquiteto de Software do IRT (Instituto Recôncavo de Tecnologia) para explorar como a portabilidade e reusabilidade de software tem sido utilizada no mercado.

P:**Boa tarde, você poderia falar um pouco da sua empresa de você e do seu cargo,de suas funções?**
R:Meu nome é Tiago, Tiago Cajahyba, trabalho no IRT empresa voltada a inovação, instituto sem fins lucrativos, não é uma empresa que visa lançar produto no mercado, na verdade nossa função é desenvolver coisas que, por exemplo, o mercado nunca viu, hoje a gente trabalha com empresas como a Samsung, sms, o grupo positivo, e outra empresas do estado de São Paulo.Minha função aqui é arquiteto de sistemas, estou aqui há 5 anos, basicamente minha função é resolver problemas.

P: **Vc poderia falar brevemente sobre como vc aplica a reusabilidade aqui?**
R: A reusabilidade está meio no cerne do que nós fazemos aqui. A primeira coisa que acontece é: Não tem nada. Não adianta fazer reuso de nada porque não tem nada. Por exemplo uma nova solução pra um celular que não foi lançado, ou seja o celular nunca foi pro mercado e ninguém sabe como ele é, chega pra gente o protótipo e temos que desenvolver soluções pra ele. Nunca fizemos aquele código. Não existe nada em que se basear. Então nesse momento não há possibilidade de reuso e o que nós fazemos: trabalhamos com esquema de prova de conceito. Criamos vários conceitos e a partir daí começamos a ter código, e dessa prova de conceito nascem novos conceitos, pois é como se fosse um grande sistema em que existem várias idéias que quando prontas, são apresentadas ao cliente e partindo desse encontro, olhamos nosso código e começamos a procurar quais as características comuns que nós temos ali e então começa a tentativa de criar pequenos componentes. As vezes nascem classes utilitárias, as vezes nascem classes de pattern ... a gente nunca sabe. Por exemplo, tratamento de imagem, que a gente trabalha muito, identificação de rosto, leitura de código de barra, identificação de objeto dentro de uma imagem e então analisamos: "bom, dentro dessas estruturas o que nós temos?".

Verificar a saturação da imagem. Bom, verificar a saturação de uma imagem é uma coisa comum, toda vez que trabalhamos com uma imagem temos que verificar: qual é o esquema de cores, qual o tipo, se é jpeg, se é bitmap, fazer histograma... e assim sucessivamente. Então essa

funções são genéricas. Desse padrão começa a nascer a nossa primeira classe utilitária. Mas estou falando só de utilitária, as vezes tambem temos regras de negócio. Por exemplo, trabalhando com celulares a gente tem o tratamento de mensagens, a gente tem um sistema android e aí chega um sms. Sempre trava alguma coisa da aplicação. Isso já tem um pouco de regra de negócio, começamos a pegar essa regra de negócio pra tratamento de exceção e organizar, pois já não é utilitário. Assim começam a nascer os novos componentes.

Passsamos 1 ano e meio mais ou menos com o projeto Android mas hoje nós temos uma API interna nossa. E é um código que permite que vc acesse basicamente todos os recursos principais do celular como contatos, sms, imagem, música tudo isso já é um componente nosso e vários projetos já estão utilizando. Também utilizamos classes de teste unitário pra garantir que tudo vai funcionar da forma esperada. O resultado tem sido muito bom, mas demorou um ano e meio pra que a gente ganhasse maturidade suficiente no código e pudesse agora utilizar em vários lugares. Vamos identifiticando o que pode ser melhorado, aparece um novo projeto, então esse novo projeto traz alguma característica. Por exemplo aconteceu agora com o facebook, o cliente queria fazer um projeto que conectasse direto com o facebook. Bom, toda essa parte de conexão é comum a vários projetos. Agora sugiu um segundo projeto que o cliente quer que use o facebook: por que não transformar aquilo que antes era apenas um código do programa em algum componente que possa ser reutilizado? Como agora na nossa API a gente já tem uma estrutura, a gente tá portando esse código pra dentro dessa estrutura e todo mundo vai poder usar. Então é assim que nasce. Não é uma coisa mirabolante, não é uma coisa fácil de fazer não é uma coisa que alguém goste de fazer até porque javadoc que tem que ter nesses casos embora não seja uma coisa muito gostosa de ser feita por parte do desenvolvedor.

P:**Quais os métodos e ferramentas de reusabilidade você trabalha mais freqüentemente?**
R:Para análise do código ou identificação dos pontos q podem sofrer reuso?
P:**Identificação dos pontos, em geral mesmo**
R:Bom, para a identificação dos pontos de reuso nós temos uma ferramenta, que agora... eu já uso há tanto tempo que esqueci o nome, ou é PMD, CheckStyle, FindBugs e CAP, acho que é criadouds, ele monta um gráfico que ele diz o quanto de Control C, Control V, tem no código,depois eu pego o nome para você, ele mostra esse código está exatamente igual a esse, se isso acontece então é porque provavelmente aquele ponto é boa sugestão de resuo ou então criação de alguma estrutura que a gente possa utilizar depois, a segunda coisa quando ele tem o nível de acoplamento do código, se a classe está muito acoplada, a gente pode verificar se há necessidade de se dividir aquele ponto em vários pedaços e mais genéricos, a partir daí a gente verifica, se esses pedaços são realmente mais genéricos?eu posso utilizar em outros lugares outros projetos?Normalmente no nosso caso aqui sim, porque Android não tem muito para onde ir, mas isso é um forte candidato a ser utilizado como reuso, e agente também utiliza algumas ferramentas de Code Smell verificar boas práticas inclusive nos nossos componentes, tipo essa API que comentei , a gente roda de tempos em tempos essa ferramenta para verificar se existe alguma coisa que possa ser melhorada.Então basicamente é assim que a gente trabalha hoje.

P:**Qual o impacto da reusabildiade na empresa?**

R:Qual o impacto que eu gostaria(riso), ou qual o impacto que realmente tem?Porque nós trabalhamos com múltiplos clientes e cada cliente tem um contrato de confidencialidade, olha a dificuldade, então o que eu faço aqui teoricamente nenhum outro cliente pode saber, então eu posso falar do meu projeto que não é pequeno,meu cliente é Samsung, nesse caso existem vários projetos, então a reusabilidade dentro do projeto é grande, mas na empresa não é tão grande, o que acontece muito é quando a gente,por exemplo, tirando Android no caso, em outros projetos que trabalhei, a gente passar, como é Java, por algumas experiências e sair, conversando com outras equipes para verificar como é que eles resolveram aquele problema, e montando uma pequena base de conhecimento, a partir daí a gente gera um código, não gosto de chamar de API nesse caso, nem de Framework, não chega no nível de Framework o que é meio comum, ai a gente gerou o Jar com varias classes extremamente comuns na base de projetos, a gente utiliza até para o Android, porque tem classe tratamento de data,classe de tratamento, leitura de arquivos, pasta de XML, assim sucessivamente, então todo mundo consegue utilizar, o legal quando é que quando um acha um erro, corrige e disponibiliza para os outros, nesse caso a reusabilidade é boa, no caso do projeto Android, inicialmente não tinha reutilização nenhuma, era um terror, e hoje nós já estamos no segundo projeto que utiliza a mesma API, sendo que existem mais três projetos, que tem previsão de utilizar a inércia, então até o final do ano teremos cinco projetos, na mesma API, eu acho isso bem interessante porque, o primeiro projeto foi muito lento, demorou um ano e um/dois mês, e segundo saiu em três meses, tem um que está entrando agora, está utilizando que já tem um mês está praticamente finalizado, então, veja é muito importante que a gente tenha como mensurar, estou já estou alertando meu gerente, olha já está podendo abaixar o prazo porque a gente investiu nisso, porque há um gosto de tempo, há um gasto de dinheiro, prefiro chamar de investimento mas há um gasto e um desgaste porque a equipe não quer usar, então a gente precisa ter alguém que compre a idéia, e fale não, eu assumo a responsabilidade, disso dar certo ou errado, hoje em dia isso não é muito fácil de encontrar, mas acho que é papel também do arquiteto, se der certo tudo bem, se não der certo amém, faz parte, mas acho que a empresa não vê muito bem dessa forma não.

P:**Qual o Framework você utiliza como mais freqüência, e por quê?**
R:Bom, quando eu trabalhava para a interprise, ou seja, desenvolvendo aplicações da empresa, eu ultiliza muito o Spring, eu usava o Spring porque o J2EE, o antigo, não era muito bom dava muito trabalho, e ninguém merece aquilo,precisava de um servidor de aplicação e assim sucessivamente, e eu queria só um Tomcat e o Spring tinha uma série de recursos muito bons, como gestão independentes, tratamento de transação, desacoplamento total do código, forçava você a trabalhar com boas práticas, e só não fazia chover ali dentro, mas tudo bem, mas o Spring era principal para a parte de negocio, e o Hibernate, não gostava muito sou da velha guarda, para projetos médios bom framework de persistência, e para interface, muito Java Script basicamente isso, agora no Android não tem muito para onde correr, tem a infra-instrutora do Android que você é obrigado a utilizar, o que nós fizemos não é um Framework , uma API, forma de acessar os recursos só, no Android tem um problema, você trabalha muito com cursor de banco de

dados, se você nunca trabalhou com banco de dados, vai fazer cada query linda, então o tratamento de transação ô meu Deus, entao o que nós fizemos transformou todo que era banco em objeto, mas veja eu não estou persistindo objeto, como o Hibernate faz, eu engano mas eu não persisto ele, eu realmente trabalho com o banco de dados, mas o programador só vê o objetos, então os objetos você não precisa criar estão todos prontos, a única coisa que você faz é conhecer os objetos, e conhecer as funções e os restante, ele não faz tudo mas ajuda bastante.

P:**Você já pegou alguma resistência para utilização da reusabilidade?**
R:(Riso)Sim claro, estagiário é um bicho muito bom, precisa ser educado, ainda bem que quando você chega em um nível e você manda, no nivel que você manda a resistencia é menor, mas ainda existe, mas agora falando serio.É um problema grave que nós temos, a reusabilidade é bonita na teoria, na pratica exige mudança de comportamento da pessoa, antigamente a pessoas eram muito mais curiosa do que são hoje, porque não tinha nada pronto, hoje em dia tem um bilhão de Framework de tudo quanto é tipo para achar no mercado, então para quê eu vou fazer?,então ai já começa o problema de reusabilidade, eu quero um framework, só que as vezes você não precisa de um canhão para matar uma formiga as vezes um pequeno código ou um pequeno conjunto de classes resolve todo seu problema, não é questão de querer reinventar a roda, é saber em que momento você deve reutilizar uma coisa que você já usou fez para outros, ou reutilizar um código que é seu, mas o programador adora fazer seu próprio código porque o código dele é sempre o melhor que existe, então que eu falo olhe use isso aqui, o principal problema é o seguinte eu dou uma função que já faz chover para pessoa, faz tudo que ele quer, mas só que ele não lê a documentação ele não pergunta programador não gosta de perguntar, há porque eu não achei eu fiz a minha, ai começa a fazer duplicação de nome de código, que é exatamente o que eu queria matar no inicio do processo, então esse é um tipo de resistência, outro tipo de resistência chave é o nível de maturidade que você quer para seu código quando você vai começar a identificar os problemas que você tem, e as soluções genéricas para ele você tem um tempo de maturação naquilo, está aprendendo as vezes a equipe zerou e chega todo mundo novo você tem que treinar aquilo, então a gerencia quer para ontem, as vezes você precisa de um tempo para treinar aquele pessoal, etc, é o que eu falei alguém precisa comprar a idéia e você tem que trabalhar com os dois lados, com aquele que não quer usar, e você vai ter que descobrir porque ele não quer usar porque as vezes o motivo de não querer usar é um bom motivo para mudar o código, e a parte da gerencia que não quer que você utilize porque quer velocidade porque não verdade você esta ganhando velocidade em um momento para perder no final porque não vai estar testado, vai dar muito mais erros quando der erro você não vai saber onde é então é uma via de duas mãos você perde mais tempo, digamos assim, no inicio do projeto porque você está estruturando bem o código, e ganha no final, porque você não vai ter dez milhões de dores de cabeça para corrigir bug's q não deveriam estar lá, e ai aparece segundo novo projeto que era as mesmas características daquele ou então a versão 2.0, iai?

P:**Quais são os obstáculos para reusabilidade ser utilizada em larga escala na empresa, e no mundo?**

R:Pô no mundo é brincadeira (riso) né?Mas tudo bem, na empresa, visão.Não a critica, visão mesmo, significa o quê, ter alguém q se preocupe com aquilo e olhe para aquilo, hoje não tem ninguém olhando para isso cabe ao arquiteto ou programador mais experiente comprar a idéia e levar até o fim, e depois falar está vendo deu certo, e depois aparece para a gerencia, quando aparece para a gerencia, as mil maravilhas porque nós fizemos isso porque nós fizemos aquilo, quando na verdade do início do projeto você estava levando porrada para fazer um código, que você sabia que era importante mas que ninguém estava acreditando, aconteceu isso, eu estava fazendo, tentando criar uma estrutura com uma outra pessoa que fosse genérico suficiente que, em vários projetos pudesse ser utilizado, porque eu sabia que aquilo seria necessário mas dava erro, obvio que dava erro, porque estava criando aquilo junto com o projeto, ninguém queria usar porque "não prestava" e ai a medida que o projeto foi evoluindo aquilo começou a prestar, daí o pessoal começou a ver que o trabalho era mais fácil, porque a única coisa que eles precisava fazer era, olha preciso de um método que faça isso ou eu dizia não o método tal já faz isso se você fizer dessa forma correto eu acrescento,mas eu ja tinha toda uma infra- estrutura montada acrescentar mais um método não era um bicho de sete cabeças, então o que falta é visão alguém que compre a idéia, alguém que fale olha, normalmente da diretoria, mas faça isso é importante ou então faça com isso aqui vou gerar dez mil projetos então o pessoal é muito imediatista, eu acho que brasileiro é muito imediatista, por causa da época da inflação ninguém fazia poupança, hoje ninguém faz poupança também, quer resolver problema logo mas não pensa daqui há um projeto, dois projetos, não vou botar tempo, vou botar projetos, daqui há um, dois projetos, como é que eu vou estar, eu acho q é isso q está faltando.No mundo é difícil , difícil porque? Existem vários tipos de mundo, por exemplo Europa é uma realidade Estados Unidos é uma realidade, Índia e Rússia, meu deus do céu aquilo ali é realidade completamente diferente, mas o que eu vejo é a Índia está correndo atrás disso, porque eu sei? Porque a maior partes dos componentes visuais que nós vemos hoje na web são feitos por programadores indianos, então se você pegar esses componentes e arrumar notará q são indianos então eu vejo eles estão se preocupando pelo menos na parte visual em criar componentes, e percebo muito que as grandes empresas como oracon e microsoft se preocupam muito com a reusabilidade mas também eles vendem ferramentas para isso, no Brasil eu vejo por exemplo o Serpro, o Serpro criou um framework chamado Demoiselle que na verdade eu não considero bem um framework , é um conjunto de framework unidos que tem uma forma de trabalhar ali e tem uma IDE mais ou menos montada no Eclipse que permite você trabalhe com eles ao mesmo tempo e uma forma de trabalhar, tem dado certo é um projeto que o governo está forçando as empresas que estão trabalhando com eles a utilizar esse framework tem muita coisa de reuso ali, então já é uma outra iniciativa.Tem muito framework nacional, Mentawai, eu lembro do Mentawai, é um framework Demoiselle também que nasceu aqui no Brasil, a idéia era não utilizar XML na configuração, eu me lembro quando o cara lançou no Java Bahia, olha eu quero fazer isso, o pessoal censurou, foi um sucesso, existem muitas frentes interessantes, o mundo caminha para isso, porque programador bom é programador preguiçoso, esse negocio de reinventar a roda é uma coisa muito antiga, então vou torcer que a gente continue nessa direção.

P:**E no mercado soteropolitano?**
R:No mercado soteropolitano, tirando Maker que há de padecer, digo que há um dia de ser um sucesso, difícil.Eu não vou chamar Dot Net de reutilização, não é porque você tem o IDE que você arrasta e solta componente, de reuso até porque não é nosso, a gente trabalha com JSS, muito com Dot Net, mas eu não vejo, tirando o Demoiselle, que o pessoal aqui do Serpro trabalha forte, eu não vejo muitas empresas se esforçando nisso, existe uma empresa chamada Avacis, quando saí de lá o pessoal estava montando uma estrutura lega, sei que a antiga Unitech,CPM Braxis, tem o seu próprio também.É bom, é, mas é limitante também então tem que saber ponderar, eu não vejo nada que chame muita atenção aqui no mercado.

P:**Você acha que a academia enfatiza reusabilidade da maneira que deve ser feita?**
R:Claro que não, a academia mal fala de reusabilidade, a academia nem usa reusabilidade, bem, na academia boa parte das coisas que a gente aprende é para que a gente tenha uma noção daquilo, no mercado de trabalho é muito diferente, não é uma critica a academia, é impossível na academia ensinar a gente a trabalhar, o mercado é mutante, quando eu entrei na academia, na minha época da faculdade estava morrendo o desktop, para nascer a web, hoje está morrendo a web para nascer dispositivos móveis, então assim gente, não é tanto tempo assim não passou tanto tempo, uma década, uma coisa mudou completamente mudou tudo, está indo para Cloud Computing, como é que a academia vai ensinar Cloud?Me diga?Tudo é no servidor, tudo é em algum lugar que você não sabe, o máximo que ela pode fazer é sistemas distribuídos porque está todo distribuído, você não sabe onde está o importante é que você saiba onde perguntar, e alguém vai lhe responder, então você aprende o conceito de sistemas distribuídos, para engenharia de software talvez, engenharia de software, acho q a academia pode ajudar bastante, porque não é tanta coisa assim, não aparece milhares de tecnologias o tempo todo, tem a CMMI, MPS.BR, BIOS, White paper, para gestão de infra-estrutura, mas para reusabilidade o máximo que pode fazer, é "Quais são os tipos de reusabilidade?Existem métodos?Existem métricas?", legal é bom aprender, na prática você não vai usar isso, porque o mercado brasileiro engatinha a gente não tem ainda maturidade suficiente.Quantas empresas CMMI você já ouviu falar que o Brasil tem nível 5?A CPM Braxis reza a lenda, é, mas é aonde?Porque você pode tirar uma área, não é a empresa inteira. Na Índia tem uma cacetada, na
Rússia tem, nos Estados Unidos tem então poxa, a gente está engatinhando. Mas isso é problema nosso?Claro que é problema nosso, sabe o que é isso, falta de visão, quantas empresas estão preocupadas com a qualidade do código que estão entregando?Nenhuma, porque o prazo é arrojado você tem profissionais que ganham muito mal, porque o mercado de Salvador não é parâmetro para nenhum lugar do país, e eu espero que a gente um dia melhore isso, mas é triste porque é um profissional altamente capacitado, ninguém faz faculdade de informática porque não gosta, porque são umas das faculdades que mais tem evasão, você vê que as faculdades de salvador todas estão com uma quantidade de alunos muito baixa, exceto as duas públicas que não contam, mas curso da Católica abre, mas ninguém se cadastra, Unifacs pouquíssimas pessoas, Rui Barbosa ainda está um pouquinho, isso porque tenho amigos que me falam isso, FTC mal as pernas. Porque está assim?Paga pouco estuda muito, você vai passar o resto da sua vida

estudando, é uma faculdade difícil, a quantidade de matéria de cálculo que a gente tem de física a depender da faculdade, é coisa de maluco, e outra coisa você vai ser taxado como nerd, resolver todos os problemas da tua família de computador, então é uma profissão dura, trabalha muito, ganha pouco e não é reconhecido. Dai você pergunta e a reusabilidade do código?(risos)Precisa de gente com visão, e nosso mercado não tem.

P:

Quais desses conceitos vc utiliza com freqüência, padrões de projeto, desenvolvimento baseado em componentes, framework de aplicação, interface com sistemas de legados, sistemas orientados a serviços, linha de produto de aplicação, integração de cots(sistemas gerais), aplicações verticais configuráveis, biblioteca de programas, gerador de programas, desenvolvimento e software orientado a aspectos, linguagem de scripts e outros?

R:

Padrões de projeto, todos os dias da minha vida e acho que aqui é necessário, e é um assunto que deveria fazer parte, uma matéria na faculdade só sobre isso,e veja, existe milhões de padrões de projeto,não é só GoF o J2EE trouxe vários, há mas agora já é JEE, mas muitos ficaram e alguns ficaram mas mudaram de nome, a idéia é a mesma quando você aprender um padrão, são coisas assim que eu acredito, tem padrão de projeto de SOA para SOA, acerta serviços, milhares de coisas, deveria ter uma matéria na faculdade só para isso porque é o dia-a-dia do programador, ontem mesmo estava atendendo uma pessoa que estava com um problema de projeto, ele foi falando o problema, já tinha nascido na minha cabeça um padrão de projeto para ele, olha isso aí é o padro de projeto mais ridículo que existe que é Factory Method, ninguém faz isso mais, parecia uma outra forma que não é GoF, mas na minha opinião é melhor, porque tem isso padrão de projeto nasceu numa época, os do GoF, que eles precisavam fazer daquela forma, hoje tem framework, as linguagens, muitas implementam padrões internos, você precisa saber, e se ele tivesse visto só aquele padrão de projeto, ele já teria resolvido o problema que ele passou dois dias para encontrar a solução, então assim é um Factory Method, é ridículo,então complicado. Desenvolvimento baseado em componentes, empresa não sempre, eu muito porque sou arquiteto é minhas responsabilidade, então sim. Framework de aplicação, todos os dias, não trabalho mais sem framework, até quando não tem eu faço, mas é necessário. Interface com sistemas de legados, claro, no momento para Android não, porque está nascendo agora, mas eu já trabalhei em outros projetos, sistemas legados é um filho que você tem que criar até a maturidade. Sistemas orientados ao serviços, nunca trabalhei, na pós eu vi mas não conheço quem trabalhe com isso a não ser o pessoal da CPM Braxis. Integração de COTS não. Aplicações verticais configuráveis também não. Biblioteca de programa, claro. Gerador de Programas, sim, mas não é total, não usa MDA, a gente gera parte do programa para facilitar a nossa vida, só algumas classe, algumas coisas que a gente já tem no padrão e a gente manda gerar, mas não é MDA. Aspectos a gente usou em alguns projetos mas em Android não é uma boa prática por motivos de restrição de memória e outra coisa, o Android já faz uma segunda compilação, ele compila para .class e depois ele compila de novo para .dex, então se eu gerar um aspecto eu terei um intermediário, e mais um intermediário, então o código final não vale a pena. Linguagem de Script, a gente usa JSP. Outros nós usamos a “carniça” do Ruby on Rails, eu não gosto, mas eu

uso, eu precisei aprender; PHP é a que vai para o céu, eu adoro, e tem gente que usa Python, tem algumas pessoas aqui que usam Python, mas basicamente JSP. Já mexemos com Lua;trabalhamos com Lua para desenvolvimento para POS, para ponto de venda, leitor de cartão de crédito e ali o software que roda embarcado em alguns clientes, como Veriphone, é feito em Lua. É uma linguagem excelente por sinal.

P:**Quais conceitos de arquitetura você mais utiliza e qualquer analista deveria saber?**

R:Conceitos de arquitetura, agora você me matou. Se for os conceitos que falamos aqui: Padrões de Projeto(ninguém é arquiteto sem saber padrões de projeto). Existe algumas coisas que a gente precisa saber: levantamento de requisitos. Tem que saber levantar requisitos. Parece fácil, não! Não é fácil. "Ah qualquer um consegue", não, ninguém consegue fazer direito. Nem o cara mais experiente. Porque? por que requisito depende de uma coisa inconstante chamada cliente.Então o máximo que a gente faz é reduzir o grau de risco que seu projeto possui.Quanto mais experiente é o programador for, mais próximo do cliente. Então, saber levantar requisitos é fundamental. Dialogar com o cliente é fundamental. Eu acho que esse diálogo marca o arquiteto. Pois mesmo que ele não converse direto com o cliente, ele vai conversar com alguém que tenha os requisitos. Então é preciso saber o que precisa ser feito. Padrões de Projeto depois. UML hoje para quem trabalha orientado a objeto é fundamental. E quando falo UML não é fazer "casinho" de uso, pq diagrama de caso de uso não é nada. Eu sei que tem gente vai me dizer; mas o que serve mesmo é a descrição do caso de uso que é um saco de fazer, mas é importante. Diagrama de sequência para quem faz componente é fundamental;no dia a dia, algumas coisas são importantes. Diagrama de atividade é bom para mostrar para o cliente. Estado em alguns momentos. Classes não precisa dizer que é fundamental. Algumas políticas de qualidade de código seria muito bom saber, mas aí depende da linguagem de programação. Então, é claro que uma ferramenta que analisa o código é muito interessante: JUnit para testes unitários de maneira geral. Pode ser com Mock, pode ser com JUnit.

P:**Você trabalha com muita refatoração aqui?**

R: Muito, oxe!! Com certeza.A gente trabalha com o método interativo incremental, então para cada interação a gente refatora código. Você vai me perguntar, que ferramenta a gente usa. Eclipse, graça a deus! Mas com alguns plugins do Android. A gente nem usa o ADT puro, a gente usa o MotoDev. Plugin da Motorola que acrescenta algumas coisas a mais. Ajuda muitas coisas nas nossas vidas. Mas refatoração, normal, normal.

P:**Na sua experiência que caso de reusabilidade você quer enfatizar? Um que marcou mesmo a sua vida.**

R: Olha, o que tá mais latente assim na minha mente é esse último caso dessa API.O que aconteceu? A verdade foi que a gente tinha um grande desafio pela frente.A gente começou a desenvolver um projeto. O projeto começou a evoluir; o código-fonte impraticável, por que? Por que o cliente mudava toda hora. O nível de refactoring chegou ao extremo. Eu não conseguia mais mudar sem quebrar tudo e aí tomei a decisão junto com uma equipe de criar um componente para resolver aquele problema que ninguém aguentava mais. O Android é uma péssima divisão de camadas. Todo mundo que fazer tudo em uma camada só. Por performance,

blá blá, por que é um celular, não tem tantos recursos de memória. Aí eu tive que comprar uma briga: “Vai fazer!”, “Vai ter três camadas!”, tudo do jeito...”Ah mais a quantidade de classes”, “Dane-se”, “Vai fazer e acabou!”. Foi uma briga durante três meses.Ninguém querendo usar e eu forçando a barra.O que aconteceu? Depois de tres meses, chegamos a maturidade daquele código. E depois de três meses todo mundo sabia onde ocorria a “zorra” daquele erro.Pronto, a partir daí todo mundo comprou a minha idéia. E hoje a gente desenvolve
muito mais, não é pouco não . Por que para mim, 40% mais rápido é muito tempo. É uma diferença absurda. E é mais ou menos 40% . Por que? por que tá tudo pronto. Ah, o cliente agora quer que meixa nas funcionalidades do evento. Ok, tudo bem.É só mexer nos componentes.Agora olha só para isso. Agora é só pedir que ele informa por você. Vc diz: Olha, eu quero entrar.Beleza.Nâo foi um projeto, já é o terceiro. Então vale a pena, e isso então é muito gratificante.

P:**Qual uma experiencia ruim de reusabilidade?**

S: Ah tem sim. Uma experiencia bastante ruim.Basicamente quando a gente trabalha com com código dos outros sempre dá algum problema por que a gente não tá acostumado. Quando eu cheguei aqui o tinha um framework(API) . Que facilitar vai ajudar a vida da gente. O negócio é que chega em um ponto. As tecnologia s suas . Não adianta ter nada se não evolui. As tecnologias evoluem. Chega no ponto que quebrou . Ou seja, tava dando erro, não tinha mais ninguém para dar manutenção. E agora? Tem que restruturar tudo. Vai todo o trabalho de novo. Isso é muito ruim. Emtão assim, não é que a reusabilidade seja ruim. É por que precisa evoluir sempre. Acabei de pensar no ponto ruim.Quando você cria a biblioteca. Ela serve para vc resolver um conjunto Xs de problemas;Não todos os problemas da humanidade.E as vezes quando você tá utilizando, você quer continuar aquilo. aí você não pode mais, como foi esse caso.Então você deve ponderá quando tem que reutilizar e quando não tem que reutilizar.Se não, digamos, fica muito mais lento a performance. Pois aquilo é muito genério, mas na verdade você quer alguma coisa pequena.

P:**Qual bom caso que não deve se usar reusabilidade de forma nenhuma?**

R:Nesse exato momento eu não tô pensando em nada.

P: Algum programa específico ou especializado que você precisou fazer reusabilidade nele.Um caminho bem ruim.

R:Eu acho que sempre é possível utilizar reusabilidade. O problema é de quê? e em que momento?Eu não consigo ver nada que não precise.Um tipo de projeto ou de problema que não precise trabalha de reusabilidade. Eu mesmo não consigo. Pode não ser em componente que você tem no momento. Mas talvez faça aquele projeto, que dali nasceu algo que pode ser reutilizado.Como foi o caso do facebook, que fizeram uma coisa completamente separada que a gente precisou: “Hum, legal” Como podemos criar um componente reutilizável.? Aí faz.Então, eu não vejo nada que não possa reutilizar.

P: **Teve algum caso que foi elaborado um código específico, otimizado. E que precisou sofrer algum problema.;**

R:Trabalha muito com sistemas interativos leva a gente a uma eterna refatoração.Chega em um ponto vc
Que vai desenvlver para dispositivos moveis(tablet também) e que você tem que escovar bit, ae cai na quele caso. O que é que eu faço? eu faço uma função de acesso direto.Que vai resolver em tempo de performance mas naum é nenhum pouco fácil de entender como aquilo usa e utilizado. Então é melhor fazer uma coisa mais genérica e que é mais fácil de ser utilizada.Por que você tem que entender que algum coitado irá ler aquele código no futuro e da manutenção. Aí cai no dilema: reutiliza ou não reutiliza? Então, é complicado. Já tive trauma: preciso aquele código que está impactando . E aí? Antes tava funcionando o habitual. Várias vezes já quebrou, váaarias vezes. Por isso que os teste unitários são fundamentais para quem trabalha com reuso.

P:**Diga aí um pattner que é o seu preferido e por que?**

R:Eu gosto de tantos. Tem que ser Gof(GAMMA, 2000). Bussiness Object. E o DAO. Por que quem trabalha com um sistema de mais de uma camada provavelmente vai passa por esses dois. Ele vai criar o seu Façade, pensando que é Façade, de repente cai no BO. Milagre. Acontece uma mágica, o Façade se tornar um BO. E o DAO, bem, não só para banco, mas para persistência em XML.E para serialização de objetos.Ele é fantástico. Quem pensou nisso vai para o céu. E os padrões de projeto que acho que são gambiarra: Adapter. Adapter não é uma coisa muito bonita não. Apesar do pessoal do GoF tenha feito.Eu respeito.É uma gabiarrazinha bonita. Arrumada. E o Singleton. Singleton também não é coisa legal. Se bem que multithread em java é difícil.Apesar de ter aprendido em java a instância única. Então eu parei de dizer que não existia Singleton multithread. Bussiness Delegate. Dispatch.Tem um monte. Os do GoF utilizamos no dia-a-dia.

P:**Poderia falar um pouco do seu currículo**

R: Me formei pela UCSAL em bacharelado em Informática,Pós-graduado pela FGV em Administração de Empresas, Pós-graduado pela Ruy Barbosa em Componentes Web, Mestrado no SENAI em Modelagem Computacional.

Referência

BOOCH, Grady. . **Object-Oriented Analysis Ad Design With Applications**. 2ed. . Massachusett:Addison Wesley Longman Limited.1998.

BRAUDE, Eric J. . **Projeto De Software: Da Programação À Arquitetura: Uma Abordagem Baseada Em Java .** Porto Alegre, RS:Bookman, 2005.

BURLAMAQUI, A. M. F. **N2N: Uma Plataforma Para Desenvolvimento De Sistemas Colaborativos Distribuídos**.Natal, RN.2004

FOWLER, Martin**. Refatoração:Aperfeiçoado O Projeto De Código Existente**. São Paulo:Bookman.2004

FOWLER2 , Martin. **Padrões de Arquitetura de Aplicações Corporativas**. São Paulo: Bookman.2006.

GAMMA, Erich; HELM, Richard; JOHNSON, Ralph;VLISSIDES, John..**Padrões De Projeto: Soluções Reutilizáveis De Software Orientado A Objetos**. São Paulo: Bookman. 2000

KERIEVSKY, Joshua**. Refatoração Para Padrões**. São Paulo: Bookman.2008.

KNUTH, D.E.. **The Art Of Programming: Fundamental Algorithms**.Massachusett :Addsion-Wesley.1971

MCCONNELL, S. **Code Complete: A Pratical Handbook Of Software Construction**. Redmond: Microsoft Press, 1993.

METSKER, S. J. **Design Patterns Java Workbook**. Massachusett:Addison-Wesley, 2002

QUINÁIA, Marcos A..**Contribuição A Uma Metodologia Para Identificação E Especificação De Padrões Arquiteturais De Software**.UTFPR:Rio Grande do Sul.Curitiba.2005.

SHALLOWAY, Alan; TROTT, Jame R**.. Explicando Padrões de Projeto: Uma nova perspectiva em projeto orientado a objeto**. São Paulo: Bookman. 2004.

SOMMERVILLE, Ian. .**Engenharia de Software**. 8 ed.Rio de Janeiro: A. Wesley, 2007.

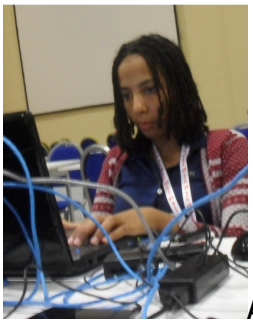
Autores

Caio Costa (nextc3@gmail.com) é graduando em Sistemas de Informação na Universidade do Estado da Bahia(UNEB). Graduando em Análise e Desenvolvimento de Sistemas no Instituto Federal de Educação, Ciência e Tecnologia da Bahia(IFBA).Bolsista do Laboratório de Realidade Aumentada, Jogos Digitais e TV Digital(LABRAGAMES).

George Dias (geocdias@gmail.com) é graduando em Sistemas de Informação na Universidade do Estado da Bahia, atua como voluntário na Consult Júnior Empresa Junior de Informática da UNEB.

 **Lauriza Santos** (laurizass@gmail.com) é graduando em Sistemas de Informação na Universidade do Estado da Bahia, estagiária na Caixa Econômica Federal.

Barbara Aniele (barbara.aniele@gmail.com) é graduando em Sistemas de Informação na Universidade do Estado da Bahia.

 **Ayran Costa** (ayranccruz@gmail.com) é graduando em Sistemas de Informação na Universidade do Estado da Bahia, é bolsista do Núcleo de Arquitetura de Computadores e Sistemas Operacionais(Acso).

Daniele Guimarães (danitgb@gmail.com) é graduando em Análise de Sistemas, trabalha na Zcr Informática.