Gmail by Google

**Fabrício Cabral <fabriciofx@gmail.com>**

# Associações bidirecionais

57 messages

**Fabrício Cabral** <fabriciofx@gmail.com> Tue, Mar 29, 2011 at 3:36 AM
To: UML-BR@yahoogrupos.com.br

Olá todos,

eu não sei se o que eu vou perguntar está on-topic, mas caso
não esteja, peço que me desculpem e ignorem a mensagem, ok? :)

Minha dúvida é a respeito das associações (ou relacionamentos)
bidirecionais. Segundo um levantamento (não aprofundado) que
eu fiz na Internet, existem as seguintes abordagens:

- Verificar se *realmente* há a necessidade deste tipo de associação
e usá-la com cuidado;

ou

- Modificar este tipo de associação para uma associação unidire-
cional.

Segundo que eu apurei, o objetivo de modificar uma associação
bidirecional é diminuir a complexidade do sistema e minimizar
um possível ciclo entre as suas classes.
Inclusive, parece que até o DDD prega também a modificação
da associação bidirecional para a unidrecional.

Assim, gostaria de saber entre os amigos aqui da lista, o que eles
acham deste assunto e como eles lidam com a questão das
associações bidirecionais em seus sistemas.

Agradeço a atenção.

--
--fx

---

**Walter Mourão** <walter.mourao@yahoo.com> Tue, Mar 29, 2011 at 8:01 AM
Reply-To: UML-BR@yahoogrupos.com.br
To: uml-br@yahoogrupos.com.br

Olá,
de forma geral não vejo nenhum problema em associações bi-direcionais... já ouvi esse papo de complexidade muitos anos atrás, referente a questões de performance nos primeiros ORM. Não acho que se aplique atualmente, pelo menos não em caráter genérico.

Sds,

Walter

[Quoted text hidden]
>[As partes desta mensagem que não continham texto foram removidas]
>

---

**Rodolpho Ugolini** <rodolpho.ugolini@gmail.com> Tue, Mar 29, 2011 at 8:50 AM
Reply-To: UML-BR@yahoogrupos.com.br
To: UML-BR@yahoogrupos.com.br

Olá Fabrício,

É On Topico total! Como assim, ignorar uma ótima pergunta?!?!

Bom, como o Walter falou, houve muita melhoria e otimização nas questões de desempenho nas linguagens/plataformas OO.

Mas discordando: como a associação bidirecional ENTRE CLASSES não se aplica somente a Frameworks de mapeamento Objeto-Relacional, e sim a todo o modelo de objetos, é sim uma boa prática evita-la sempre que possível.

A razão é simples: otimizar os recursos de máquina disponíveis. Assim como vivemos numa era de abundância de recursos computacionais, também vivemos em uma época de demandas altamente volateis, onde podemos ir de dezenas a milhões de chamadas de uma hora para outra.

Explico: uma classe "Endereço" que é usada hoje em um sisteminha de cadastro de clientes interno, acessado por dezenas de pessoas, pode ser reutilizada e passar a fazer parte do sistema de venda online da empresa que será acessado por milhares ou até milhões de pessoas. Se esta classe tiver uma associação bidirecional com a classe "Estado", todas as referencias (ponteiros) de Estado para Endereço também teriam de ser mantidas em memória. Numa situação limite, o sistema ao carregar um único endereço, teria de carregar a classe estado e por tabela todos os endereços deste estado. Como normalmente, não precisamos saber os endereços do Estado em um sistema de venda on-line, poderia neste caso ser uma associação unidirecional de "endereço" para "estado", salvando desta forma muitos ciclos de processamento e memória.

Portanto, é sim uma boa prática "enxugar" o modelo ao final de cada iteração, procurando manter associações bidirecionais somente onde se mostrar necessário.

Vamos ver o que o pessoal acha!

Grande abraço e parabéns pela pergunta!

Rodolpho
[Quoted text hidden]

UML-BR-unsubscribe@yahoogrupos.com.br

<*> O uso que você faz do Yahoo! Grupos está sujeito aos:
http://br.yahoo.com/info/utos.html

---

**RUTHEMBERG JÚNIOR** <ruthemberg@bol.com.br> Tue, Mar 29, 2011 at 10:48 AM
Reply-To: UML-BR@yahoogrupos.com.br
To: UML-BR@yahoogrupos.com.br

Vamos lembrar do velho chavão UML:
"Precisamos modelar O QUÊ o sistema deve fazer e não COMO o sistema deve fazer/funcionar.
Por essa frase, eu te respondo, SIM!!! Use bi-direcionais sem dó!
Se o sistema precisa e a modelagem premite, então mãos à obra.
Gargalos, concorencias, etc... cabem a profissionais de Infra que devem saber o que é preciso para atender o sistema modelado.

Em resumo, o que eu quero dizer é que a tecnologia está aí para ser usada, mas se ela será usada ou não cabe a nós decidirmos.

[]´s

RUTHEMBERG JÚNIOR
Consultor de TI
Analista de Sistemas
cel 19 97136252

[Quoted text hidden]
[Quoted text hidden]



---

**Thiago Nascimento** <nascimenthiago@gmail.com> Tue, Mar 29, 2011 at 11:10 AM
Reply-To: UML-BR@yahoogrupos.com.br
To: UML-BR@yahoogrupos.com.br

Experiência própria: bi-direcionamento usado irrestritamente levou a sistemas de e-gov não funcionarem. O grafo de objetos utilizado tornou-se gigantesco e inviabilizou o atendimento de um número considerável de requisições.

Conclusão: Bi-direciomento deve ser pensado e usado quando necessário estritamente necessário.

2011/3/29 RUTHEMBERG JÚNIOR <ruthemberg@bol.com.br>

>
>
> Vamos lembrar do velho chavão UML:

> "Precisamos modelar O QUÊ o sistema deve fazer e não COMO o sistema deve
> fazer/funcionar.
> Por essa frase, eu te respondo, SIM!!! Use bi-direcionais sem dó!
> Se o sistema precisa e a modelagem premite, então mãos à obra.
> Gargalos, concorencias, etc... cabem a profissionais de Infra que devem
> saber o que é preciso para atender o sistema modelado.
>
> Em resumo, o que eu quero dizer é que a tecnologia está aí para ser usada,
> mas se ela será usada ou não cabe a nós decidirmos.
>
> []´s
>
> RUTHEMBERG JÚNIOR
> Consultor de TI
> Analista de Sistemas
> cel 19 97136252
>
> Em 29/03/2011 08:50, Rodolpho Ugolini < rodolpho.ugolini@gmail.com >
[Quoted text hidden]
--
Thiago Nascimento
perl -e '$_="tMM naaCt Feocmama_itpUilucoGa";$_.=$1,print $2 while
s/(..)(.)//;print substr$_,1,1;'

[As partes desta mensagem que não continham texto foram removidas]

---

Links do Yahoo! Grupos

[Quoted text hidden]

---

**Fabrício Cabral** <fabriciofx@gmail.com> Tue, Mar 29, 2011 at 11:27 AM
To: UML-BR@yahoogrupos.com.br

Olá Rodolpho, tudo bem?

2011/3/29 Rodolpho Ugolini <rodolpho.ugolini@gmail.com>

> Olá Fabrício,
>
> É On Topico total! Como assim, ignorar uma ótima pergunta?!?!

Ah, que bom! Tive receio que não fosse... :)

> Bom, como o Walter falou, houve muita melhoria e otimização nas questões de
> desempenho nas linguagens/plataformas OO.
>
> Mas discordando: como a associação bidirecional ENTRE CLASSES não se aplica
> somente a Frameworks de mapeamento Objeto-Relacional, e sim a todo o modelo
> de objetos, é sim uma boa prática evita-la sempre que possível.
>
> A razão é simples: otimizar os recursos de máquina disponíveis. Assim como
> vivemos numa era de abundância de recursos computacionais, também vivemos em
> uma época de demandas altamente volateis, onde podemos ir de dezenas a
> milhões de chamadas de uma hora para outra.
>
> Explico: uma classe "Endereço" que é usada hoje em um sisteminha de cadastro
> de clientes interno, acessado por dezenas de pessoas, pode ser reutilizada e
> passar a fazer parte do sistema de venda online da empresa que será acessado
> por milhares ou até milhões de pessoas. Se esta classe tiver uma associação

> bidirecional com a classe "Estado", todas as referencias (ponteiros) de Estado para Endereço também teriam de ser mantidas em memória. Numa situação limite, o sistema ao carregar um único endereço, teria de carregar a classe estado e por tabela todos os endereços deste estado. Como normalmente, não precisamos saber os endereços do Estado em um sistema de venda on-line, poderia neste caso ser uma associação unidirecional de "endereço" para "estado", salvando desta forma muitos ciclos de processamento e memória.

Me corrija se eu estiver enganado, mas o Lady Loading [1] (Carregamento Tardio), que a grande maioria dos ORM implementam, não serve justamente para se evitar este problema? Assim, utilizando-o evitaríamos ter esta quantidade a mais de memória consumida. A modelagem continuaria com a bidirecionalidade, mas, por questão de otimização da implementação, seria aplicado o Lazy Loading.

> Portanto, é sim uma boa prática "enxugar" o modelo ao final de cada iteração, procurando manter associações bidirecionais somente onde se mostrar necessário.
>
> Vamos ver o que o pessoal acha!

Vamos! Até agora eu estou gostando do debate... :)

> Grande abraço e parabéns pela pergunta!

Obrigado! Esta (e outras tantas) é uma dúvida que eu sempre tive e até agora não tinha obtido uma resposta satisfatória... :)

Abraço!

[1] http://en.wikipedia.org/wiki/Lazy_loading

--
--fx

---

**Francis Pontes de Albuquerque** <francis@eteg.com.br> Tue, Mar 29, 2011 at 1:15 PM
Reply-To: UML-BR@yahoogrupos.com.br
To: UML-BR@yahoogrupos.com.br
Cc: RUTHEMBERG JÚNIOR <ruthemberg@bol.com.br>

Eu acredito que devemos aprofundar um pouco mais para entender qual é o contexto.

Não trabalho mais com desenvolvedor. Sei qual é a importância defendida pelo Rodolpho. E quando eu estava aprendendo, era muito bom não precisar de pensar em problemas de desempenho. Quando eu recebia o diagrama de classe sem a pessoa que o construiu tivesse pensado neste problemas de desempenho (associação bidirecional), corria o risco de desenvolver da forma "crua". E sempre eu fazia um código que nasce sem ter aplicado as boas práticas (evitar relacionamento bidirecional). Isto por que eu não tinha maturidade de perceber o erro do diagrama que estava recebendo. E por isto implementava do jeito que estava. Sem questionar.

Em uma boa parte dos casos, o problema era pequeno. Os frameworks e a "infra" davam conta do baixo desempenho que uma má decisão de associação bidirecional havia criado. E não incomodava o usuário.

Em outros casos, algumas funcionalidades críticas, era necessário reformular a associação.

Lazy Load não resolve todos os casos! Tome cuidado com isto!

Hoje trabalho mais como Analista de Requisito. E eu crio o diagrama de classe inicial. E nesta atividade, de extrema importância, concordo com o Ruthemberg no ponto de desfio de atenção ao detalhes técnicos. Se eu ficar preocupado em aplicar todas as boas práticas, corro o risco de fazer um diagrama não muito bom para o negócio.

E ai? Como fazer então?

Hoje, fazemos um único diagrama. Mas ele é criado e revisado pelo Analista, Arquiteto e Desenvolvedor. Assim conseguimos as melhores competências para construir e evoluir o digrama.

Tem um resultado melhor que na época que apenas um fazia, ou cada uma fazia o seu.

[]´s
Francis Pontes

Em 29 de março de 2011 10:48, RUTHEMBERG JÚNIOR <ruthemberg@bol.com.br>escreveu:

> Vamos lembrar do velho chavão UML:
> "Precisamos modelar O QUÊ o sistema deve fazer e não COMO o sistema deve
> fazer/funcionar.
> Por essa frase, eu te respondo, SIM!!! Use bi-direcionais sem dó!
> Se o sistema precisa e a modelagem premite, então mãos à obra.
> Gargalos, concorencias, etc... cabem a profissionais de Infra que devem
> saber o que é preciso para atender o sistema modelado.
>
> Em resumo, o que eu quero dizer é que a tecnologia está aí para ser usada,
> mas se ela será usada ou não cabe a nós decidirmos.
>
> []´s
>
> RUTHEMBERG JÚNIOR
> Consultor de TI
> Analista de Sistemas
> cel 19 97136252 begin_of_the_skype_highlighting 19 97136252
> end_of_the_skype_highlighting begin_of_the_skype_highlighting
> 19 97136252 begin_of_the_skype_highlighting 19
> 97136252 end_of_the_skype_highlighting
> end_of_the_skype_highlighting

[Quoted text hidden]



---

**Fabrício Cabral** <fabriciofx@gmail.com> Tue, Mar 29, 2011 at 2:02 PM
To: UML-BR@yahoogrupos.com.br

Olá Francis, tudo bem?

2011/3/29 Francis Pontes de Albuquerque <francis@eteg.com.br>

> Eu acredito que devemos aprofundar um pouco mais para entender qual é o contexto.
>
> Não trabalho mais com desenvolvedor. Sei qual é a importância defendida pelo Rodolpho. E quando eu estava aprendendo, era muito bom não precisar de pensar em problemas de desempenho. Quando eu recebia o diagrama de classe sem a pessoa que o construiu tivesse pensado neste problemas de desempenho (associação bidirecional), corria o risco de desenvolver da forma "crua". E sempre eu fazia um código que nasce sem ter aplicado as boas práticas (evitar relacionamento bidirecional). Isto por que eu não tinha maturidade de perceber o erro do diagrama que estava recebendo. E por isto implementava do jeito que estava. Sem questionar.
>
> Em uma boa parte dos casos, o problema era pequeno. Os frameworks e a "infra" davam conta do baixo desempenho que uma má decisão de associação bidirecional havia criado. E não incomodava o usuário.

Me desculpe a pergunta "trivial" mas, por que uma associação bidirecional é uma má decisão? Gostaria, se possível, de uma literatura que corroborasse com esta afirmativa.

> Em outros casos, algumas funcionalidades críticas, era necessário reformular a associação.
>
> Lazy Load não resolve todos os casos! Tome cuidado com isto!

Você poderia citar um exemplo que o Lazy Load não resolveria?

> Hoje trabalho mais como Analista de Requisito. E eu crio o diagrama de classe inicial. E nesta atividade, de extrema importância, concordo com o Ruthemberg no ponto de desfio de atenção ao detalhes técnicos. Se eu ficar preocupado em aplicar todas as boas práticas, corro o risco de fazer um diagrama não muito bom para o negócio.
>
> E ai? Como fazer então?
>
> Hoje, fazemos um único diagrama. Mas ele é criado e revisado pelo Analista, Arquiteto e Desenvolvedor. Assim conseguimos as melhores competências para construir e evoluir o digrama.
>
> Tem um resultado melhor que na época que apenas um fazia, ou cada uma fazia o seu.

Francis, agradeço muito você ter compartilhado sua experiência conosco, mas eu ainda tenho algumas dúvidas. Pelo que eu entendi, você afirma que as associações bidirecionais são uma má escolha. No entanto, gostaria (se possível) de um esclarecimento acerca de como tratá-las. Devo retirá-las de deixar apenas uniderecionais? E nos casos onde além da bidirecionalidade também há multiplicidade N:M entre as classes? Como resolver?

Atenciosamente,

--
--fx

---

**Rodolpho Ugolini** <rodolpho.ugolini@gmail.com> Tue, Mar 29, 2011 at 2:03 PM

Reply-To: UML-BR@yahoogrupos.com.br
To: UML-BR@yahoogrupos.com.br

Ruthemberg,

Acho que você se confundiu: a frase refere-se a Casos de Uso... não a UML.

A UML pode (e deve) modelar o COMO de um sistema sim: enquanto os casos de uso refletem o "o quê" o sistema faz, os modelos de análise e design descrevem o "como".

E me desculpe discordar de voce de novo, mas cabe a nós, SIM, decidirmos como a tecnologia vai ser usada...

Abraço,
Rodolpho

2011/3/29 RUTHEMBERG JÚNIOR <ruthemberg@bol.com.br>

> 
> 
> Vamos lembrar do velho chavão UML:
> "Precisamos modelar O QUÊ o sistema deve fazer e não COMO o sistema deve
> fazer/funcionar.
> Por essa frase, eu te respondo, SIM!!! Use bi-direcionais sem dó!
> Se o sistema precisa e a modelagem premite, então mãos à obra.
> Gargalos, concorencias, etc... cabem a profissionais de Infra que devem
> saber o que é preciso para atender o sistema modelado.
> 
> Em resumo, o que eu quero dizer é que a tecnologia está aí para ser usada,
> mas se ela será usada ou não cabe a nós decidirmos.
> 
> []´s
> 
> RUTHEMBERG JÚNIOR
> Consultor de TI
> Analista de Sistemas
> cel 19 97136252
> 
> Em 29/03/2011 08:50, Rodolpho Ugolini < rodolpho.ugolini@gmail.com >
[Quoted text hidden]
[Quoted text hidden]

---

**Marcelo Elias Del Valle** <mvallebr@gmail.com> Tue, Mar 29, 2011 at 2:08 PM
To: UML-BR@yahoogrupos.com.br
Cc: Fabrício Cabral <fabriciofx@gmail.com>

Olá Fabrício,

Primeiramente, gostaria de concordar com o Rodolpho. A pergunta é totalmente on-topic, gostaria que fizessem mais perguntas parecidas na lista.
Segundo, sobre a bidirecionalidade, evite mesmo sempre que possível. A palavra mágica aqui é ACOPLAMENTO.
Quando você cria uma associação unidirecional da classe A para a classe B, está criando também uma relação de dependência, ou seja, só conseguirá usar A se tiver B junto. Contudo, a classe B continua "independente", de forma que você conseguiria isolá-la no seu sistema se isso fosse útil ou necessário. Quando você cria uma associação bidirecional, não consegue mais separá-las, vai sempre depender de ambas em qualquer situação em que precise de uma.

Abraços,
Marcelo.

Em 29 de março de 2011 11:27, Fabrício Cabral <fabriciofx@gmail.com> escreveu:

> Olá Rodolpho, tudo bem?
>
> 2011/3/29 Rodolpho Ugolini <rodolpho.ugolini@gmail.com>
> [Quoted text hidden]
> [Quoted text hidden]
> --fx
>
> [As partes desta mensagem que não continham texto foram removidas]
>
> 

--
Marcelo Elias Del Valle
http://SorocabaMundi.com - Sorocaba Mundi - Tudo relacionado a Sorocaba
http://www.humansoftware.com.br

---

**Fabrício Cabral** <fabriciofx@gmail.com> Tue, Mar 29, 2011 at 2:38 PM
To: UML-BR@yahoogrupos.com.br

Olá Marcelo,

2011/3/29 Marcelo Elias Del Valle <mvallebr@gmail.com>

> Olá Fabrício,
>
> Primeiramente, gostaria de concordar com o Rodolpho. A pergunta é totalmente on-topic, gostaria que fizessem mais perguntas parecidas na lista.

Eu também gostaria. Principalmente aquelas que aparentemente são "óbvias" ou aquelas que todo mundo sabe a resposta mas não sabe explicar o porquê da resposta.

> Segundo, sobre a bidirecionalidade, evite mesmo sempre que possível. A palavra mágica aqui é ACOPLAMENTO.
> Quando você cria uma associação unidirecional da classe A para a classe B, está criando também uma relação de dependência, ou seja, só conseguirá usar A se tiver B junto. Contudo, a classe B continua "independente", de forma que você conseguiria isolá-la no seu sistema se isso fosse útil ou necessário. Quando você cria uma associação bidirecional, não consegue mais separá-las, vai sempre depender de ambas em qualquer situação em que precise de uma.

Ah, agora sim eu entendi por que evitá-las. :) Mas e como tratá-las? Tirar a bidirecionalidade "à força" e deixar a associação unidirecional? Se sim, como saber qual lado (ou classe) é que deve ficar com a referência a outra classe? E nos casos onde não dá pra evitar a bidirecionalidade, o que fazer para diminuir o acoplamento? Implementar interfaces para que as associações sejam direcionais às interfaces e não às classes?

Agradeço a atenção!

--
--fx

---

**Rafael Chaves** <rchaves.jug@gmail.com> Tue, Mar 29, 2011 at 2:41 PM
Reply-To: UML-BR@yahoogrupos.com.br
To: UML-BR@yahoogrupos.com.br

Vou discordar da tua discordancia, Rodolpho!

Diagramas de classe sao usados tanto em analise como em projeto (design).

Analise se preocupa com o problema sendo compreendido, entao concordo com o Ruthemberg que em um modelo de analise, se conceitualmente faz sentido, uma associacao direcional deve ser usada.

Projeto se preocupa com a solucao, e como mencionado por ti e outros, concordo que associacoes bidirecionais devam ser usadas com parcimonia.

Finalizando, notem que o fato de uma associacao ser modelada em uma unica direcao nao necessariamente proibe que haja navegacao na direcao oposta. Pode apenas sugerir que navegar na direcao oposta seja *mais caro*. O significado exato (proibicao/sugestao) depende do contexto/decisao do time/etc.

My CAD$ 0,02.

Rafael
http://abstratt.com/blog/

2011/3/29 Rodolpho Ugolini <rodolpho.ugolini@gmail.com>

> Ruthemberg,
>
> Acho que você se confundiu: a frase refere-se a Casos de Uso... não a UML.
>
> A UML pode (e deve) modelar o COMO de um sistema sim: enquanto os casos de
> uso refletem o "o quê" o sistema faz, os modelos de análise e design
> descrevem o "como".
>
> E me desculpe discordar de voce de novo, mas cabe a nós, SIM, decidirmos
> como a tecnologia vai ser usada...
>
> Abraço,
> Rodolpho
>
> 2011/3/29 RUTHEMBERG JÚNIOR <ruthemberg@bol.com.br>
>
> >
> >
> > Vamos lembrar do velho chavão UML:
> > "Precisamos modelar O QUÊ o sistema deve fazer e não COMO o sistema deve
> > fazer/funcionar.
> > Por essa frase, eu te respondo, SIM!!! Use bi-direcionais sem dó!
> > Se o sistema precisa e a modelagem premite, então mãos à obra.
> > Gargalos, concorencias, etc... cabem a profissionais de Infra que devem
> > saber o que é preciso para atender o sistema modelado.
> >
> > Em resumo, o que eu quero dizer é que a tecnologia está aí para ser
> usada,
> > mas se ela será usada ou não cabe a nós decidirmos.

> >
> > []´s
> >
> > RUTHEMBERG JÚNIOR
> > Consultor de TI
> > Analista de Sistemas
> > cel 19 97136252
> >
> > Em 29/03/2011 08:50, Rodolpho Ugolini < rodolpho.ugolini@gmail.com >

[Quoted text hidden]



---

**RUTHEMBERG JÚNIOR** <ruthemberg@bol.com.br> Tue, Mar 29, 2011 at 2:36 PM
Reply-To: UML-BR@yahoogrupos.com.br
To: UML-BR@yahoogrupos.com.br

Rodolfo, nao esquenta, discordar é um direito seu e eu respeito.
O que eu não entendo é porque as pessoas insistem em discordar de algo que elas não entenderam (veja que na sua frase final, a que vc diz que discorda, diz exatamente o mesmo que eu disse em meu email).

Eu:
Em resumo, o que eu quero dizer é que a tecnologia está aí para ser usada, mas se ela será usada ou não cabe a nós decidirmos.

Voce:
E me desculpe discordar de voce de novo, mas cabe a nós, SIM, decidirmos como a tecnologia vai ser usada...

Outra coisa, fui generalista demais na minha resposta porque achei que todos da lista tem uma boa visão das ferramentas que usam, e quando disse que a tal frase é da UML, é porque o Caso de Uso sem UML não existe, não é verdade?

boas tardes e abraços.

[]´s

RUTHEMBERG JÚNIOR
Consultor de TI
Analista de Sistemas
cel 19 97136252

Em 29/03/2011 14:03, Rodolpho Ugolini < rodolpho.ugolini@gmail.com > escreveu:
Ruthemberg,

Acho que você se confundiu: a frase refere-se a Casos de Uso... não a UML.

A UML pode (e deve) modelar o COMO de um sistema sim: enquanto os casos de uso refletem o "o quê" o sistema faz, os modelos de análise e design

descrevem o "como".

E me desculpe discordar de voce de novo, mas cabe a nós, SIM, decidirmos como a tecnologia vai ser usada...

Abraço,
Rodolpho

2011/3/29 RUTHEMBERG JÚNIOR

> Vamos lembrar do velho chavão UML:
> "Precisamos modelar O QUÊ o sistema deve fazer e não COMO o sistema deve
> fazer/funcionar.
> Por essa frase, eu te respondo, SIM!!! Use bi-direcionais sem dó!
> Se o sistema precisa e a modelagem premite, então mãos à obra.
> Gargalos, concorencias, etc... cabem a profissionais de Infra que devem
> saber o que é preciso para atender o sistema modelado.
>
> Em resumo, o que eu quero dizer é que a tecnologia está aí para ser usada,
> mas se ela será usada ou não cabe a nós decidirmos.
>
> []´s
>
> RUTHEMBERG JÚNIOR
> Consultor de TI
> Analista de Sistemas
> cel 19 97136252
>
> Em 29/03/2011 08:50, Rodolpho Ugolini < rodolpho.ugolini@gmail.com >

[Quoted text hidden]



---

**Francis Pontes de Albuquerque** <francis@eteg.com.br> Tue, Mar 29, 2011 at 3:17 PM
To: UML-BR@yahoogrupos.com.br
Cc: Fabrício Cabral <fabriciofx@gmail.com>

Fabrício,

já lhe aviso que meu conhecimento é limitado, pois não tenho profundo conhecimento sobre o assunto.

Quem poder, favor colaborar com outros exemplos e argumentos.

Em 29 de março de 2011 14:02, Fabrício Cabral <fabriciofx@gmail.com> escreveu:

> Olá Francis, tudo bem?
>
> 2011/3/29 Francis Pontes de Albuquerque <francis@eteg.com.br>
> Me desculpe a pergunta "trivial" mas, por que uma associação bidirecional é
> uma
> má decisão? Gostaria, se possível, de uma literatura que corroborasse com

> esta
> afirmativa.

Pergunta difícil. Não sou um poço de conhecimento literário. Mas gosto de aplicar bons princípios.

Principio KISS

http://en.wikipedia.org/wiki/KISS_principle

Se o objeto A não precisa se relacionar com B, mas B precisa se relacionar com A, por que você vai fazer um relacionamento bidirecional?

Lembre-se! Não sou contra relacionamento bidirecional quando é necessário. Mas quando vejo um diagrama de classe que tem apenas relacionamento bidirecional, acredito que quem construiu não o fez com "carinho".

Outro exemplo já citado.

> "A razão é simples: otimizar os recursos de máquina disponíveis. Assim como
> vivemos numa era de abundância de recursos computacionais, também vivemos em
> uma época de demandas altamente volateis, onde podemos ir de dezenas a
> milhões de chamadas de uma hora para outra.
>
> Explico: uma classe "Endereço" que é usada hoje em um sisteminha de cadastro
> de clientes interno, acessado por dezenas de pessoas, pode ser reutilizada e
> passar a fazer parte do sistema de venda online da empresa que será acessado
> por milhares ou até milhões de pessoas. Se esta classe tiver uma associação
> bidirecional com a classe "Estado", todas as referencias (ponteiros) de
> Estado para Endereço também teriam de ser mantidas em memória. Numa situação
> limite, o sistema ao carregar um único endereço, teria de carregar a classe
> estado e por tabela todos os endereços deste estado. Como normalmente, não
> precisamos saber os endereços do Estado em um sistema de venda on-line,
> poderia neste caso ser uma associação unidirecional de "endereço" para
> "estado", salvando desta forma muitos ciclos de processamento e memória."

> >
> > Lazy Load não resolve todos os casos! Tome cuidado com isto!
> >
> Você poderia citar um exemplo que o Lazy Load não resolveria?

Quando a funcionalidade precisar de alto desempenho e baixo tempo de resposta. Exemplo: Vou precisar de um atributo do objeto A e outros atributos que estão na lista de objetos B dentro do objeto A. Deixar o lazy load "on" seria perda de tempo! No "for" da exibição da lista de objetos B, para cada objeto, vc terá que ir ao banco. Nesta caso, a quantidade de chamadas na camada de persistência seria 1 (objeto A) + n (Lista dos objetos B). Se o lazy load estiver "off", você faria apenas uma chamada à camada de persistência, buscando todos os objetos de uma só vez.

Importante notar que a vantagem (ou desvantagem) do relacionamento pode (e deve) ser questionada de acordo com a necessidade de desempenho de cada solução. Por isto é difícil responder sua pergunta, na lata! Depende muito do contexto de utilização do sistema.

Os avanços dos frameworks de persistência parecem resolver o problema, fazendo boa otimização no uso de cache e layload.

Não sou especialista. Não vou conseguir aprofundar muito. Mas há um outro bom argumento.

Se lazy load fosse bom para aplicar em todos os casos, não existiria a possibilidade de ligar ou desliga-lo. Ele seria sempre "on" 8-)

> Hoje trabalho mais como Analista de Requisito. E eu crio o diagrama de
> classe inicial. E nesta atividade, de extrema importância, concordo com o
> Ruthemberg no ponto de desfio de atenção ao detalhes técnicos. Se eu ficar
> preocupado em aplicar todas as boas práticas, corro o risco de fazer um
> diagrama não muito bom para o negócio.
>
> E ai? Como fazer então?
>
> Hoje, fazemos um único diagrama. Mas ele é criado e revisado pelo Analista,
> Arquiteto e Desenvolvedor. Assim conseguimos as melhores competências para
> construir e evoluir o digrama.
>
> Tem um resultado melhor que na época que apenas um fazia, ou cada uma fazia
> o seu.
>

Francis, agradeço muito você ter compartilhado sua experiência conosco, mas eu
ainda tenho algumas dúvidas. Pelo que eu entendi, você afirma que as associações
bidirecionais são uma má escolha. No entanto, gostaria (se possível) de um esclarecimento acerca de como tratá-las. Devo retirá-las de deixar apenas uniderecionais? E nos casos onde além da bidirecionalidade também há multiplicidade
N:M entre as classes? Como resolver?

Deixo para nossos amigos da lista com maior experiência.

[]´s
Francis

---

**Rafael Chaves** <rchaves.jug@gmail.com> Tue, Mar 29, 2011 at 3:42 PM
Reply-To: UML-BR@yahoogrupos.com.br
To: UML-BR@yahoogrupos.com.br

Adicionando: um modelo de analise poderia simplesmente se abster de definir direcionalidade, se nao for relevante para compreender o dominio em questao.

2011/3/29 Rafael Chaves <rchaves.jug@gmail.com>

[Quoted text hidden]



---

**Marcelo Elias Del Valle** <mvallebr@gmail.com> Tue, Mar 29, 2011 at 4:12 PM
To: UML-BR@yahoogrupos.com.br
Cc: Fabrício Cabral <fabriciofx@gmail.com>

Fabrício,

Vamos ver se consigo responder. :D

*"Mas e como tratá-las? Tirar a bidirecionalidade "à força" e deixar a associação unidirecional? Se sim, como saber qual lado (ou classe) é que deve ficar com a referência a outra classe?*
*E nos casos onde não dá pra evitar a bidirecionalidade, o que fazer para diminuir o acoplamento? Implementar interfaces para que as associações sejam direcionais às interfaces e não às classes?"*

Na verdade, a pergunta que sempre temos que fazer quando estamos realizando o design é: **por quê**? Por que essa referência circular está aí? Por que eu tive de criar uma relação bidirecional. Esse é o melhor modo de tratar, questionando o seu próprio design. A associação bidirecional pode existir porque você criou uma dependência entre as classes sem necessidade. Existe dependência de negócio?

Exemplo: tenho uma classe Pessoa e uma Telefone. Eu crio uma relação unidirecional de pessoa para telefone (Pessoa.getTelefones()) porque preciso saber acessar os telefones de uma pessoa. Pessoa não estaria completa sem seus telefones, ela depende dos telefones.

Nessa única relação, já posso questionar: faria sentido pra mim, em termos de negócio, tentar separar Pessoa de Telefone? Eu teria mais reuso de pessoa? Se em metade das regras de negócio do seu sistema você precisa acessar telefone sempre que tenta acessar uma Pessoa, não faz sentido tentar separar, deixe acoplado mesmo e assuma que Pessoa depende de telefone. Contudo, se Pessoa acessa telefone mas fizer sentido para a regra que ela dependa de algo abstrato, como itens possuídos, você pode criar uma interface ItemPossuido a qual é implementada por Telefone. Pessoa então teria uma dependência por itens possuídos, sejam telefones ou quaisquer outras classes que implementem essa interface.

Suponhamos agora que você precise saber quantas pessoas possuem um mesmo número de telefone (imagine que o mesmo número é usado por várias pessoas) e para isso crie uma associação de Telefone para Pessoa (Telefone.getPessoas()). A associação ficou agora bi direcional. Por quê? A regra de várias pessoas possuírem um telefone faz sentido se Telefone for uma classe independente? Eu poderia substituir Pessoa por uma interface e obter algum reuso? Talvez se eu colocar um atributo a mais em Telefone dizendo quantos utilizadores ele tem deixe telefone mais reutilizável e independente (embora isso sugira uma desnormalização do modelo). Talvez não, talvez não faça o menor sentido que telefone seja reutilizável e essas duas entidades tenham sempre que existir juntas.

**Em resumo: depende do negócio. Desacoplar é bom, mas tem seu custo. Por vezes, temos que deixar as coisas acopladas mesmo. Vale a pena quebrar uma dependência SEMPRE?**

Abraços,
Marcelo.

Em 29 de março de 2011 14:38, Fabrício Cabral <fabriciofx@gmail.com> escreveu:

> Olá Marcelo,
>
> 2011/3/29 Marcelo Elias Del Valle <mvallebr@gmail.com>
> [Quoted text hidden]
> [Quoted text hidden]
>
> --
> --fx
>
> [As partes desta mensagem que não continham texto foram removidas]

> __,_._,___

--
Marcelo Elias Del Valle
http://SorocabaMundi.com - Sorocaba Mundi - Tudo relacionado a Sorocaba
http://www.humansoftware.com.br

---

**Ivan Carvalho** <ivanjpc@gmail.com> Tue, Mar 29, 2011 at 5:14 PM
Reply-To: UML-BR@yahoogrupos.com.br
To: UML-BR@yahoogrupos.com.br

Só para correção. É LAZY Load (e não Lady Load).

Eu tenho feito bastante modelagem usando bidireção como valor default.
Entendo que o grafo de dependência é um fator de risco enorme para o projeto
e por isto, ao realizar a contrução, o valor da bidireção é sempre lazy.
neste momento é feito uma análise da necessidade de não ser Lazy.

Em 29 de março de 2011 11:27, Fabrício Cabral <fabriciofx@gmail.com>escreveu:

>
>
> Olá Rodolpho, tudo bem?
>
>
[Quoted text hidden]
[Quoted text hidden]

---

**Marcelo Elias Del Valle** <mvallebr@gmail.com> Tue, Mar 29, 2011 at 6:34 PM
To: UML-BR@yahoogrupos.com.br
Cc: Fabrício Cabral <fabriciofx@gmail.com>

Fabrício,

Só para fortalecer um pouco mais o argumento, transcrevo aqui o último parágrafo da página http://domaindrivendesign.org/resources/what_is_ddd :

"*Yet the most significant complexity of many applications is not technical. It is in the domain itself, the activity or business of the user. When this domain complexity is not dealt with in the design, it won't matter that the infrastructural technology is well-conceived. A successful design must systematically deal with this central aspect of the software.*"

Abraços,
Marcelo.
[Quoted text hidden]

---

**Charles Abreu** <charlesabreu@gmail.com> Wed, Mar 30, 2011 at 10:16 PM
To: UML-BR@yahoogrupos.com.br
Cc: Fabrício Cabral <fabriciofx@gmail.com>

Oi, Fabrício.

Deixe-me tentar dar minha contribuição prática.

Do ponto de vista de análise, a minha recomendação é sempre usar relacionamentos bidirecionais. No mundo real uma coisa se liga à outra e vice-versa, não é? Porque não seria igual em um modelo do mundo real? É simples: pense em todas as associações como bidirecionais. Isso torna o modelo muito mais simples de se lidar, e as regras e as restrições se tornam evidentes e intuitivas.

Na implementação, vai depender da tecnologia usada. Para quem usa o Hibernate, a minha recomendação é sempre manter todas as associações bidirecionais e lazy. Sempre. Essa é a graça do Hibernate: ele te permite mapear sua implementação mais próxima do modelo de análise.

Há duas coisas nessas minhas recomendações que as pessoas costumam questionar, como outros colegas sabiamente já fizeram entre si. A primeira é: "Mas o relacionamento bidirecional não aumenta o acoplamento? A questão do 'fraco acoplamento e forte coesão" não é justamente contrário a isso?'" Sim e não. O problema é que a expressão "fraco acoplamento e forte coesão" está incompleta, dita assim. Falta um elemento nessa ideia: o domínio (ou módulo). Um domínio é um conjunto coeso de conceitos referentes a um aspecto do problema. Cliente e Endereço podem pertencer ao mesmo domínio, mas ProdutoEmEstoque e FilaJMS certamente pertencem a domínios diferentes.

O domínio é a unidade mínima de reutilização de código; esse é um princípio em algumas metodologias OO. E ele leva a uma consequência muito importante: se o domínio é indivisível e suas classes não devem ser utilizadas separadamente, não há razão para nos preocuparmos com dependências cíclicas entre elas. Pelo contrário, elas são desejáveis, pois demonstram a coesão entre elas. Por outro lado, se os domínios são reutilizáveis, devemos mantê-los os mais independentes entre si que pudermos. Ou seja, devemo reduzir o acoplamento entre eles. Assim, podemos reescrever a expressão original da seguinte forma: "fraco acoplamento inter-domínios e forte coesão intra-domínio". E, assim, nossos relacionamentos bidirecionais intra-domínio não estão mais violando nenhuma boa prática.

A segunda questão é quanto à performance da implementação. Repetindo, isso depende da tecnologia usada. No caso do Hibernate, não há problema, desde que todas as associações sejam lazy. Nos pontos em que precisar carregar os relacionamentos em uma tacada só, simplesmente adicione um "fetch join" na query. Assim estão em produção vários sistemas, alguns com volume de acessos e dados de causar espanto em muita gente experiente. Para outras tecnologias, é preciso avaliar.

Espero ter ajudado mais do que atrapalhado.

Charles Abreu
http://twitter.com/#!/charlesabreu
https://pagseguro.uol.com.br
http://www.vousossegado.com.br

2011/3/29 Fabrício Cabral <fabriciofx@gmail.com>

> [Quoted text hidden]
> [As partes desta mensagem que não continham texto foram removidas]
>
> ___.__,__.___
>
> | **através de email** | Responder **através da web** | **Adicionar um novo tópico**
> Mensagens neste tópico (**1**)
>
> **ATIVIDADE NOS ÚLTIMOS DIAS:** **Novos usuários 2**
> Visite seu Grupo
>
> **Tem muita gente querendo te conhecer! Que tal dar uma chance?**
>
> Yahoo! Grupos
>
> Trocar para: SÃ³ Texto, Resenha DiÃ¡ria • Sair do grupo • Termos de uso
>
> ___,__.__,___

---

**Marcelo Elias Del Valle** <mvallebr@gmail.com> Wed, Mar 30, 2011 at 10:45 PM
To: UML-BR@yahoogrupos.com.br
Cc: Charles Abreu <charlesabreu@gmail.com>, Fabrício Cabral <fabriciofx@gmail.com>

Fabrício,

Eu estava tentando dizer mais ou menos a mesma coisa que o Charles, mas acho que ele foi muito mais feliz na explicação. Basicamente, o que você considera um domínio depende do negócio. Se você enxergar que vale a pena desacoplar, deixe em domínios separados. Se você escolheu deixar no mesmo domínio, não vale a pena ficar tentando tirar a bidirecionalidade, já que o próprio negócio pede acoplamento. Novamente ressalto minha frase: não existe sistema mais correto, existe sistema que melhor atende à necessidade de negócio.

Charles,

Só discordo de uma coisa que você disse, do fato de "sempre usar os relacionamentos bidirecionais". Quando você utiliza Classes que só tem dados para representar o banco, isso pode funcionar bem, com lazy loading. Mas se você estiver usando active record, por exemplo, colocar associações desnecessariamente não seria interessante. Você pode muito bem começar um projeto achando que suas 8 classes pertencem ao mesmo domínio e no meio dele descobrir que pode valer a pela deixar 5 em um e 3 em outro, concorda? Eu acho que só devemos usar a bidirecionalidade quando o negócio realmente pede.

Abraços,
Marcelo.

Em 30 de março de 2011 22:16, Charles Abreu <charlesabreu@gmail.com> escreveu:

> Oi, Fabrício.
>
> Deixe-me tentar dar minha contribuição prática.
>
> Do ponto de vista de análise, a minha recomendação é *sempre usar relacionamentos bidirecionais*. No mundo real uma coisa se liga à outra e vice-versa, não é? Porque não seria igual em um modelo do mundo real? É simples: pense em todas as associações como bidirecionais. Isso torna o modelo muito mais simples de se lidar, e as regras e as restrições se tornam evidentes e intuitivas.
>
> Na implementação, vai depender da tecnologia usada. Para quem usa o Hibernate, a minha recomendação é *sempre manter todas as associações bidirecionais e lazy*. Sempre. Essa é a graça do Hibernate: ele te permite mapear sua implementação mais próxima do modelo de análise.
>
> Há duas coisas nessas minhas recomendações que as pessoas costumam questionar, como outros colegas sabiamente já fizeram entre si. A primeira é: "Mas o relacionamento bidirecional não aumenta o acoplamento? A questão do 'fraco acoplamento e forte coesão" não é justamente contrário a isso?'" Sim e não. O problema é que a expressão "fraco acoplamento e forte coesão" está incompleta, dita assim. Falta um elemento nessa ideia: o domínio (ou módulo). Um domínio é um conjunto coeso de conceitos referentes a um aspecto do problema. Cliente e Endereço podem pertencer ao mesmo domínio, mas ProdutoEmEstoque e FilaJMS certamente pertencem a domínios diferentes.
>
> O domínio é a unidade mínima de reutilização de código; esse é um princípio em algumas metodologias OO. E ele leva a uma consequência muito importante: se o domínio é indivisível e suas classes não devem ser utilizadas separadamente, não há razão para nos preocuparmos com dependências cíclicas entre elas. Pelo contrário, elas são desejáveis, pois demonstram a coesão entre elas. Por outro lado, se os domínios são reutilizáveis, devemos mantê-los os mais independentes entre si que pudermos. Ou seja, devemo reduzir o acoplamento entre eles. Assim, podemos reescrever a expressão original da seguinte forma: "fraco acoplamento *inter-domínios* e forte coesão *intra-domínio*". E, assim, nossos relacionamentos bidirecionais intra-domínio não estão mais violando nenhuma boa prática.
>
> A segunda questão é quanto à performance da implementação. Repetindo, isso depende da tecnologia usada. No caso do Hibernate, não há problema, *desde que todas as associações sejam lazy*. Nos pontos em que precisar carregar os relacionamentos em uma tacada só, simplesmente adicione um "fetch join" na query. Assim estão em produção vários sistemas, alguns com volume de acessos

> e dados de causar espanto em muita gente experiente. Para outras
> tecnologias, é preciso avaliar.
>
> Espero ter ajudado mais do que atrapalhado.
>
> Charles Abreu
> http://twitter.com/#!/charlesabreu
> <http://twitter.com/#!/charlesabreu>https://pagseguro.uol.com.br
> <https://pagseguro.uol.com.br/>http://www.vousossegado.com.br
> [Quoted text hidden]
> ---
>
> Links do Yahoo! Grupos
>
> <*> Para visitar o site do seu grupo na web, acesse:
> http://br.groups.yahoo.com/group/UML-BR/
>
> <*> Para sair deste grupo, envie um e-mail para:
> UML-BR-unsubscribe@yahoogrupos.com.br
>
> <*> O uso que você faz do Yahoo! Grupos está sujeito aos:
> http://br.yahoo.com/info/utos.html

--
Marcelo Elias Del Valle
http://SorocabaMundi.com - Sorocaba Mundi - Tudo relacionado a Sorocaba
http://www.humansoftware.com.br

---

**Fabrício Cabral** <fabriciofx@gmail.com> Thu, Mar 31, 2011 at 1:34 AM
To: UML-BR@yahoogrupos.com.br

> Olá Charles,
>
> primeiramente, gostaria de lhe agradecer pelo seu e-mail. O mesmo é muito conciso
> e detalhista. A parte do "fraco acoplamento inter-domínios e forte coesão intra-domínio"
> ,sem dúvida, é primoroso! :)
>
> É esse tipo de discussão, na minha opinião, que faz enriquecer essa lista! Parabéns a
> todos os participantes! :)
>
> Agora Charles, me permita fazer uma observação: pelo que eu entendi, você recomenda
> sempre usar os relacionamentos bidirecionais e utilizar o Hibernate com lazy em todas
> associações, correto? Mas a utilização do Hibernate com lazy não seria uma forma
> de utilizar uma tecnologia de ORM para sanar uma deficiência do modelo analisado?
> A deficiência no caso, seria o não mapeamento correto das associações que deveriam
> e as que não devam ser bidirecionais.
>
> Atenciosamente,
>
> 2011/3/30 Charles Abreu <charlesabreu@gmail.com>
> [Quoted text hidden]

--
--fx

---

**Marcelo Elias Del Valle** <mvallebr@gmail.com> Thu, Mar 31, 2011 at 1:01 PM
To: UML-BR@yahoogrupos.com.br

Cc: Fabrício Cabral <fabriciofx@gmail.com>

Fabrício,

O Charles ainda deve responder, então perdõe eu responder por cima :D
Eu acho que o uso de lazy loading tem a ver com a implementação. É uma forma de conseguir boa performance ao consultar o modelo.
Já o fato de haver ou não dependência entre as entidades, ou seja, o seu modelo, esse depende exclusivamente do negócio. Independente da sua implementação, o modelo de domínio (OO) deve ser o mesmo.

Abraços,
Marcelo.

Em 31 de março de 2011 01:34, Fabrício Cabral <fabriciofx@gmail.com> escreveu:

> Olá Charles,
>
> primeiramente, gostaria de lhe agradecer pelo seu e-mail. O mesmo é muito
> conciso
> e detalhista. A parte do "fraco acoplamento *inter-domínios* e forte coesão
> *intra-domínio*"
> ,sem dúvida, é primoroso! :)
>
> É esse tipo de discussão, na minha opinião, que faz enriquecer essa lista!
> Parabéns a
> todos os participantes! :)
>
> Agora Charles, me permita fazer uma observação: pelo que eu entendi, você
> recomenda
> sempre usar os relacionamentos bidirecionais e utilizar o Hibernate com lazy
> em todas
> associações, correto? Mas a utilização do Hibernate com lazy não seria uma
> forma
> de utilizar uma tecnologia de ORM para sanar uma deficiência do modelo
> analisado?
> A deficiência no caso, seria o não mapeamento correto das associações que
> deveriam
> e as que não devam ser bidirecionais.
>
> Atenciosamente,
>
> 2011/3/30 Charles Abreu <charlesabreu@gmail.com>
>
> > Oi, Fabrício.
> >
> > Deixe-me tentar dar minha contribuição prática.
> >
> > Do ponto de vista de análise, a minha recomendação é *sempre usar
> > relacionamentos bidirecionais*. No mundo real uma coisa se liga à outra e
> > vice-versa, não é? Porque não seria igual em um modelo do mundo real? É
> > simples: pense em todas as associações como bidirecionais. Isso torna o
> > modelo muito mais simples de se lidar, e as regras e as restrições se tornam
> > evidentes e intuitivas.
> >
> > Na implementação, vai depender da tecnologia usada. Para quem usa o
> > Hibernate, a minha recomendação é *sempre manter todas as associações
> > bidirecionais e lazy*. Sempre. Essa é a graça do Hibernate: ele te permite
> > mapear sua implementação mais próxima do modelo de análise.
> >
> > Há duas coisas nessas minhas recomendações que as pessoas costumam
> > questionar, como outros colegas sabiamente já fizeram entre si. A primeira
> > é: "Mas o relacionamento bidirecional não aumenta o acoplamento? A questão
> > do 'fraco acoplamento e forte coesão" não é justamente contrário a isso?'"
> > Sim e não. O problema é que a expressão "fraco acoplamento e forte coesão"
> > está incompleta, dita assim. Falta um elemento nessa ideia: o domínio (ou

> módulo). Um domínio é um conjunto coeso de conceitos referentes a um aspecto
> do problema. Cliente e Endereço podem pertencer ao mesmo domínio, mas
> ProdutoEmEstoque e FilaJMS certamente pertencem a domínios diferentes.
>
> O domínio é a unidade mínima de reutilização de código; esse é um princípio
> em algumas metodologias OO. E ele leva a uma consequência muito importante:
> se o domínio é indivisível e suas classes não devem ser utilizadas
> separadamente, não há razão para nos preocuparmos com dependências cíclicas
> entre elas. Pelo contrário, elas são desejáveis, pois demonstram a coesão
> entre elas. Por outro lado, se os domínios são reutilizáveis, devemos
> mantê-los os mais independentes entre si que pudermos. Ou seja, devemo
> reduzir o acoplamento entre eles. Assim, podemos reescrever a expressão
> original da seguinte forma: "fraco acoplamento *inter-domínios* e forte
> coesão *intra-domínio*". E, assim, nossos relacionamentos bidirecionais
> intra-domínio não estão mais violando nenhuma boa prática.
>
> A segunda questão é quanto à performance da implementação. Repetindo, isso
> depende da tecnologia usada. No caso do Hibernate, não há problema, *desde
> que todas as associações sejam lazy*. Nos pontos em que precisar carregar
> os relacionamentos em uma tacada só, simplesmente adicione um "fetch join"
> na query. Assim estão em produção vários sistemas, alguns com volume de
> acessos e dados de causar espanto em muita gente experiente. Para outras
> tecnologias, é preciso avaliar.
>
> Espero ter ajudado mais do que atrapalhado.
>
> Charles Abreu
> http://twitter.com/#!/charlesabreu
> <http://twitter.com/#!/charlesabreu>https://pagseguro.uol.com.br
> <https://pagseguro.uol.com.br/>http://www.vousossegado.com.br
[Quoted text hidden]
[Quoted text hidden]
---

Links do Yahoo! Grupos

<*> Para visitar o site do seu grupo na web, acesse:
http://br.groups.yahoo.com/group/UML-BR/

<*> Para sair deste grupo, envie um e-mail para:
UML-BR-unsubscribe@yahoogrupos.com.br

<*> O uso que você faz do Yahoo! Grupos está sujeito aos:
http://br.yahoo.com/info/utos.html

--
Marcelo Elias Del Valle
http://SorocabaMundi.com - Sorocaba Mundi - Tudo relacionado a Sorocaba
http://www.humansoftware.com.br

---

**Marcelo Andrade** <mfandrade@gmail.com> Thu, Mar 31, 2011 at 2:28 PM
Reply-To: UML-BR@yahoogrupos.com.br
To: UML-BR@yahoogrupos.com.br

2011/3/29 RUTHEMBERG JÚNIOR <ruthemberg@bol.com.br>:
> Vamos lembrar do velho chavão UML:
> "Precisamos modelar O QUÊ o sistema deve fazer e não COMO o sistema deve fazer/funcionar.
> Por essa frase, eu te respondo, SIM!!! Use bi-direcionais sem dó!

> Se o sistema precisa e a modelagem premite, então mãos à obra.
> Gargalos, concorencias, etc... cabem a profissionais de Infra que devem saber o que é preciso para atender o sistema modelado.

Talvez não tenha muito a contribuir. Também aprendi que o "correto" é até utilizar associações bidirecionais de início, mas com o tempo, ao refinar o modelo, deveria ser reavaliado para considerar sua real necessidade.

Por esta resposta do Ruthemberg e dos demais concluo que num modelo de análise, associações bidirecionais são toleráveis, mas deve ser refinado para verificar sua necessidade ao considerar questões de implementação no modelo de design.

Atts.

--
MARCELO F ANDRADE
Belem, Amazonia, Brazil

"I took the red pill"



---

**Marcio Tierno** <mtierno@rocketmail.com> Thu, Mar 31, 2011 at 2:43 PM
Reply-To: UML-BR@yahoogrupos.com.br
To: UML-BR@yahoogrupos.com.br

Ruthemberg

Caso de uso EXISTE sim sem UML. Inclusive, Casos de Uso foram criados pelo vovô Jacobson VÁÁÁRIOS anos antes da UML existir.

MT.

--- Em UML-BR@yahoogrupos.com.br, RUTHEMBERG JÚNIOR <ruthemberg@...> escreveu
>
> Rodolfo, nao esquenta, discordar é um direito seu e eu respeito.
> O que eu não entendo é porque as pessoas insistem em discordar de algo que elas não entenderam (veja que na sua frase final, a que vc diz que discorda, diz exatamente o mesmo que eu disse em meu email).
>
> Eu:
> Em resumo, o que eu quero dizer é que a tecnologia está aí para ser usada, mas se ela será usada ou não cabe a nós decidirmos.
>
> Voce:
> E me desculpe discordar de voce de novo, mas cabe a nós, SIM, decidirmos como a tecnologia vai ser usada...
>
> Outra coisa, fui generalista demais na minha resposta porque achei que todos da lista tem uma boa visão

das ferramentas que usam, e quando disse que a tal frase é da UML, é porque o Caso de Uso sem UML não existe, não é verdade?
>
> boas tardes e abraços.
>
>
> []´s
>
> RUTHEMBERG JÚNIOR
> Consultor de TI
> Analista de Sistemas
> cel 19 97136252
>
>
>
>
>

[Quoted text hidden]
> > Em 29/03/2011 08:50, Rodolpho Ugolini < rodolpho.ugolini@... >
[Quoted text hidden]



---

**Marcio Tierno** <mtierno@rocketmail.com> Thu, Mar 31, 2011 at 2:59 PM
Reply-To: UML-BR@yahoogrupos.com.br
To: UML-BR@yahoogrupos.com.br

Em cima das respostas de todos...

a) Esse negócio de modelo de análise e modelo de design é da época em que eu era comunista. Levante a mão quem MANTÉM os dois modelos conforme o projeto vai evoluindo. Há, isso sim, modelos de design com maior ou menor nível de abstração.

b) Uma associação bi-direcional entre a classe A e a classe B significa que na classe A haverá um ponteiro para B e na B haverá um ponteiro para A. Não só isso: significa que se um objeto a1 apontar para um objeto b1, o objeto b1 deve apontar para a1 tb. E se a1 passar a apontar para b2, b1 deverá apontar para null e b2 deverá apontar para a1 e, ainda, se algum possível a2 estiver apontando para b2, deverá passar a apontar para null. Isso é na prática o que significa uma associação bi-direcional numa multiplicidade 1 para 1, que é a mais simples. (Se você implementa diferente, está implementado *errado* - ou então deveria mudar seu modelo para refletir a realidade do seu código - no caso, duas associações unidirecionais.)

Pergunto: alguém aí vai topar ficar programando isso tudo sem que seja ABSOLUTAMENTE necessário?

É por isso que a imensa maioria das associações é unidirecional. Inclusive, programadores espertos mandam designers prolixos plantarem batatas e implementam unidirecionalidade mesmo quando o modelo define bi-direcionalidade, salvo quando a bi-direcionalidade é absolutamente necessária.

c) LAZY Load não resolve problemas de direcionalidade mas, sim, de consumo de memória por instanciação desnecessária. Você pode usar lazy load com associações bi ou unidirecionais, mas sempre de um para

muitos ou de muitos para muitos.

d) Nunca menospreze o impacto que a economia que um par de bytes e meia dúzia de ciclos de máquina pode ter sobre o funcionamento global de um sistema. Em projetos de gente grande, como sistemas bancários, de cartões de crédito ou telefonia, isso pode significar a diferença entre a rotina rodar na janela de tempo certa ou não. Para um banco, furar a janela de processamento significa não abrir as portas no outro dia. Já pensaram nisso?

e) Vocês sabem o que significa discurso bacharelesco? É o que alguns colegas usam de vez em quando. Na falta de argumentação melhor, repete-se o óbvio de maneira empolada, tentando com isso dar consistência a uma tese que carece de argumentos fundamentadores.

Por exemplo: é claro que o Rodolpho sabe que se pode navegar na "direção contrária" de uma associação unidirecional. Além do fato do cara *ENSINAR* isso na IBM/Rational há mais de 10 anos, ele também veleja e, se consegue fazer um barco a vela navegar na direção contrária do vento, certamente sabe sair de um objeto e chegar a outro no sentido contrário da direcionalidade.

[]s

MT (chutando as canelas, para variar).



---

**Marcelo Elias Del Valle** <mvallebr@gmail.com> Thu, Mar 31, 2011 at 3:27 PM
Reply-To: UML-BR@yahoogrupos.com.br
To: UML-BR@yahoogrupos.com.br
Cc: Marcio Tierno <mtierno@rocketmail.com>

MT,

> Ok o que você escreveu no post mas... Peraí, você foi comunista?!????
> Esquece tudo e explique isso agora!!! :D
>
> Não dava pra perder a piada, desculpem... ;-)

Abraços,
Marcelo.
[Quoted text hidden]
--
Marcelo Elias Del Valle
http://SorocabaMundi.com <http://sorocabamundi.com/> - Sorocaba Mundi - Tudo relacionado a Sorocaba
http://www.humansoftware.com.br

[As partes desta mensagem que não continham texto foram removidas]

<*> Para visitar o site do seu grupo na web, acesse:
http://br.groups.yahoo.com/group/UML-BR/

<*> Para sair deste grupo, envie um e-mail para:
[Quoted text hidden]

---

**Rafael Chaves** <rchaves.jug@gmail.com> Thu, Mar 31, 2011 at 4:33 PM
Reply-To: UML-BR@yahoogrupos.com.br
To: UML-BR@yahoogrupos.com.br

Modelo de análise e modelo de projeto são coisas diferentes... entender o problema não significa entender a solução. Eles podem até muitas vezes serem isomórficos, mas podem muito bem ser bastante diferentes (entidades no modelo de análise podem não existir no modelo de design, modelo de design pode introduzir abstrações que não existiam no modelo de análise). Modelo de análise mapeando 1:1 para modelo de design provavelmente significa que se está trabalhando numa aplicação CRUD simples que mesmo um Analista de Negócios/usuário avançado conseguiria conceber. Sinto muito.

Além disso, um bom AN/BA deve conseguir fazer um bom modelo de análise, até melhor que um desenvolvedor. Mas território de AN é o problema, não a solução. Território de desenvolvedor é primariamente solução (e não estou falando de tecnologia). É claro que um desenvolvedor pode criar um modelo de design tendo um modelo de análise como ponto de partida, e sair modificando ele liberalmente para atender a solução planejada.

Mudar o modelo de análise? Só se o entendimento do problema (ou o próprio problema) mudar. O entendimento da solução não tem impacto no modelo de análise (a não ser que revele deficiências no modelo de análise - ou seja, o entendimento do problema mudou).

Tendo dito tudo isso - ANs provavelmente ficariam chocados com a idéia - mas sendo um desenvolvedor, não sou radicalmente contra pular-se a fase/modelo de análise como also separado, afinal (limitação de muitos desenvolvedores) eu penso melhor em termos de solução, e cliente e usuário entende mesmo é programa rodando, e não diagrama UML.

É claro, tudo isso é opinião/valor pessoal, e não verdade universal, como tudo o que se escreve nessa lista.

Falou,

Rafael
http://abstratt.com/blog
[Quoted text hidden]
[Quoted text hidden]

---

**Fabrício Cabral** <fabriciofx@gmail.com> Thu, Mar 31, 2011 at 4:35 PM
To: UML-BR@yahoogrupos.com.br

Olá Márcio,

2011/3/31 Marcio Tierno <mtierno@rocketmail.com>

> Em cima das respostas de todos...
>
> a) Esse negócio de modelo de análise e modelo de design é da época em que eu era comunista. Levante a mão quem MANTÉM os dois modelos conforme o projeto vai evoluindo. Há, isso sim, modelos de design com maior ou menor nível de abstração.

Eu sou novato na área de análise de sistemas e uma coisa que eu sempre me pergunto é se realmente há a necessidade de haver estes dois modelos (o de análise e o de design). Mas acho que este já seria assunto pra uma outra discussão... :)

> b) Uma associação bi-direcional entre a classe A e a classe B significa que na classe A haverá um ponteiro para B e na B haverá um ponteiro para A. Não só isso: significa que se um objeto a1 apontar para um objeto b1, o objeto b1 deve apontar para a1 tb. E se a1 passar a apontar para b2, b1 deverá apontar para null e b2 deverá apontar para a1 e, ainda, se algum possível a2 estiver apontando para b2, deverá passar a apontar para null. Isso é na prática o que significa uma associação bi-direcional numa multiplicidade 1 para 1, que é a mais simples. (Se você implementa diferente, está implementado *errado* - ou então deveria mudar seu modelo para refletir a realidade do seu código - no caso, duas associações unidirecionais.)
>
> Pergunto: alguém aí vai topar ficar programando isso tudo sem que seja ABSOLUTAMENTE necessário?

Pelo seu argumento, dá pra perceber que uma associação bidirecional é BEM custosa, seja pelo trabalho que o código terá que realizar para refletir esta associação, bem como o processamento e memória que será gasto para tal tarefa. Mas isto eu acredito que a grande maioria aqui já sabia disso. O ponto em questão é: existe alguma maneira *sistemática* de identificar que uma associação não precise (ou precise) ser bidirecional?

> É por isso que a imensa maioria das associações é unidirecional. Inclusive, programadores espertos mandam designers prolixos plantarem batatas e implementam unidirecionalidade mesmo quando o modelo define bi-direcionalidade, salvo quando a bi-direcionalidade é absolutamente necessária.

Aqui eu iria analisar com cuidado. E se o programador *achar* que a bidirecionalidade não era necessária e implementar uma unidirecionalidade quando não o deveria fazer? Afinal, acredito que era papel do analista analisar o problema por todos os lados e dizer: "olha aqui realmente deveria haver uma associação bidirecional
pela razão X e Y".

Assim, conforme já havia perguntado anteriormente: haveria alguma maneira *sistemática* para determinar que uma associação deveria ou não deveria ser bidirecional?

> c) LAZY Load não resolve problemas de direcionalidade mas, sim, de consumo de memória por instanciação desnecessária. Você pode usar lazy load com associações bi ou unidirecionais, mas sempre de um para muitos ou de muitos para muitos.

Não é possível usar Lazy Load em uma associação de um para um?

> d) Nunca menospreze o impacto que a economia que um par de bytes e meia dúzia de ciclos de máquina pode ter sobre o funcionamento global de um sistema. Em projetos de gente grande, como sistemas bancários, de cartões de crédito ou telefonia, isso pode significar a diferença entre a rotina rodar na janela de tempo certa ou não. Para um banco, furar a janela de processamento significa não abrir as portas no outro dia. Já pensaram nisso?

Eu não menosprezo este impacto, mas convenhamos, que se eu fosse rodar um sistema em um domínio crítico como um banco, obviamente eu faria e refaria testes de escalabilidade antes de colocar o sistema em produção, certo? ;)

> e) Vocês sabem o que significa discurso bacharelesco? É o que alguns colegas usam de vez em quando. Na falta de argumentação melhor, repete-se o óbvio de maneira empolada, tentando com isso dar consistência a uma tese que carece de argumentos fundamentadores.

Calma que a gente chega lá. :)

> Por exemplo: é claro que o Rodolpho sabe que se pode navegar na "direção contrária" de uma associação unidirecional. Além do fato do cara *ENSINAR* isso na IBM/Rational há mais de 10 anos, ele também veleja e, se consegue fazer um barco a vela navegar na direção contrária do vento, certamente sabe sair de um objeto e chegar a outro no sentido contrário da direcionalidade.

Bom, como eu não sou o Rodolpho, eu não seu como navegar na "direção contrária" de uma associação unidirecional. Seria utilizando recursos de reflection?

> MT (chutando as canelas, para variar).

É, eu percebi... mas com calma a gente chega lá... :)

--
--fx

---

**Fabrício Cabral** <fabriciofx@gmail.com> Thu, Mar 31, 2011 at 4:39 PM
To: UML-BR@yahoogrupos.com.br

Olá Marcelo,

2011/3/31 Marcelo Andrade <mfandrade@gmail.com>

> Talvez não tenha muito a contribuir. Também aprendi que o "correto" é até utilizar associações bidirecionais de início, mas com o tempo, ao refinar o modelo, deveria ser reavaliado para considerar sua real necessidade.
>
> Por esta resposta do Ruthemberg e dos demais concluo que num modelo de análise, associações bidirecionais são toleráveis, mas deve ser refinado para verificar sua necessidade ao considerar questões de implementação no modelo de design.

Bom, isso também é o que (até onde eu pude perceber) a maioria dos livros de análise pregam. Só que até agora eu não pude capturar uma maneira *sistemática* para fazer essa "transformação", digamos assim.

Obrigado pela atenção!

--fx

---

**RUTHEMBERG JÚNIOR** <ruthemberg@bol.com.br> Thu, Mar 31, 2011 at 5:00 PM
Reply-To: UML-BR@yahoogrupos.com.br
To: UML-BR@yahoogrupos.com.br

Gente, vamos parar de procurar pelo em ovo.
Vc vive na época do velho Jaco? A propósito, o documento Caso de Uso como conhecemos hoje, foi aprimorada e proposta por Alistair Cockburn.
Poxa, além do mais, no que esse tipo de "eu sei mais do que você" contribui para questão levantada?
Esse tempo dispensado para ficar batendo de frente (sem méritos algum), seria com certeza muito melhor aproveitado pela lista se empregado na resposta e auxílio aos membros do forum.

[]´s

RUTHEMBERG JÚNIOR
Consultor de TI
Analista de Sistemas
cel 19 97136252

Em 31/03/2011 14:43, Marcio Tierno < mtierno@rocketmail.com > escreveu:

Ruthemberg

Caso de uso EXISTE sim sem UML. Inclusive, Casos de Uso foram criados pelo vovô Jacobson VÁÁÁRIOS anos antes da UML existir.

MT.

--- Em UML-BR@yahoogrupos.com.br, RUTHEMBERG JÚNIOR escreveu

[Quoted text hidden]



---

**Marcio Tierno** <mtierno@rocketmail.com> Thu, Mar 31, 2011 at 5:26 PM
Reply-To: UML-BR@yahoogrupos.com.br
To: UML-BR@yahoogrupos.com.br

Então, Rafael.

Isomórfico quer dizer "mesma forma", né? I.e., são iguais.

De novo recorrendo a Jacobson, o ponto é o seguinte: você consegue pensar só no problema mesmo? Quando se faz uma abstração da realidade para se construir um modelo, já se está pensando na solução, pois a decisão sobre quais elementos de detalhe esconder (princípio de abstração) direciona decisivamente como será a construção final.

Na maioria das vezes, o trabalho de modelagem situa-se numa zona cinzenta que pode ser melhor descrita como sendo design, mas com níveis de abstração variáveis.

Isso para não dizer que, com o advento das abordagens ágeis, essa separação toda de AN/AS/D/C/T e etc deixou de fazer sentido. Que dirá separar os modelos desse jeito.

MT.

--- Em UML-BR@yahoogrupos.com.br, Rafael Chaves <rchaves.jug@...> escreveu

[Quoted text hidden]
> http://abstratt.com/blog
>
> 2011/3/31 Marcio Tierno <mtierno@...>
[Quoted text hidden]

___,_._,___

**Marcio Tierno** <mtierno@rocketmail.com> Thu, Mar 31, 2011 at 5:27 PM
Reply-To: UML-BR@yahoogrupos.com.br
To: UML-BR@yahoogrupos.com.br

Já fui a passeata na Praça da Sé gritando "arroz, feijão, saúde e educação". kkkkkk!

(mal aí o off topic!)

MT.

--- Em UML-BR@yahoogrupos.com.br, Marcelo Elias Del Valle <mvallebr@...> escreveu
>
> MT,
>
> Ok o que você escreveu no post mas... Peraí, você foi comunista?!????
> Esquece tudo e explique isso agora!!! :D
>
> Não dava pra perder a piada, desculpem... ;-)
>
> Abraços,
> Marcelo.
>

[Quoted text hidden]
> http://SorocabaMundi.com <http://sorocabamundi.com/> - Sorocaba Mundi - Tudo
> relacionado a Sorocaba
> http://www.humansoftware.com.br
>
>
> [As partes desta mensagem que não continham texto foram removidas]
>

___.__,_.___



___,_._,___

**Marcio Tierno** <mtierno@rocketmail.com> Thu, Mar 31, 2011 at 5:36 PM
Reply-To: UML-BR@yahoogrupos.com.br
To: UML-BR@yahoogrupos.com.br

Fabricio

Vamos lá:

1) O ponto em questão é: existe alguma maneira *sistemática* de identificar que uma associação não precise (ou precise) ser bidirecional?

R: Tem. Use unidirecional por definição. Se precisar ser bidirecional, essa necessidade se fará explícita e gritante em algum ponto na codificação. Um refactoring básico e tá tudo resolvido.

2) Aqui eu iria analisar com cuidado. E se o programador *achar* que a bidirecionalidade não era necessária e implementar uma unidirecionalidade quando não o deveria fazer? Afinal, acredito que era papel do analista analisar o problema por todos os lados e dizer: "olha aqui realmente deveria haver uma associação bidirecional pela razão X e Y".

A bidirecionalidade só é necessária por questões de implementação, nunca de projeto ou análise. Se o desenvolvedor conseguir responder aos requisitos do sistema com unidirecionalidade, deve fazê-lo. Bidirecionalidade não é requisito, é ferramenta de construção.

3) > Não é possível usar Lazy Load em uma associação de um para um?

R: Possível é, só não faz sentido. Dê uma revisada no pattern e chegue à conclusão. Construir todo um aparato para carregar objetos conforme a necessidade e descobrir que há um único objeto a carregar - e que normalmente já estará carregado...

4) obviamente eu faria e refaria testes de escalabilidade antes de colocar o sistema em produção, certo? ;)

Certo. A diferença seria a quantidade de mexidas a serem feitas no sistema...

5) eu não seu como navegar na "direção contrária" de uma associação unidirecional. Seria utilizando recursos de reflection?

Tem vários jeitos. Reflection é um jeito bem sofisticado. Alguns usam apoio do banco de dados. Outros "passeiam" pelos objetos até achar seu "dono". Se a navegação no sentido contrário é muito frequente e ela começa a comer recursos importantes do sistema, eis aí a necessidade da bidirecionalidade.

Muito boas as suas perguntas e comentários. Parabéns!!!!!

MT.

--- Em UML-BR@yahoogrupos.com.br, Fabrício Cabral <fabriciofx@...> escreveu
>
> Olá Márcio,
>
> 2011/3/31 Marcio Tierno <mtierno@...>

[Quoted text hidden]
> [As partes desta mensagem que não continham texto foram removidas]
>



**Rafael Chaves** <rchaves.jug@gmail.com> Thu, Mar 31, 2011 at 6:11 PM
Reply-To: UML-BR@yahoogrupos.com.br

To: UML-BR@yahoogrupos.com.br

> Isomórfico quer dizer "mesma forma", né? I.e., são iguais.

Mesma forma (e mesmo assim, nem sempre), mas não iguais.

> você consegue pensar só no problema mesmo?
> Quando se faz uma abstração da realidade para se construir um modelo, já se está pensando na
> solução, pois a decisão sobre quais elementos de detalhe esconder (princípio de abstração) direciona
> decisivamente como será a construção final.

Concordo que qualquer modelo vai sempre omitir coisas da realidade modelada (por definição), mas alguem colhendo requisitos é capaz de saber muito bem o que é relevante sobre o problema, sem ter menor idéia do que a solução vai ser (apenas o que a solução precisa satisfazer). Concordo que o modelo de design é derivado do modelo de análise, mas essa derivacão não é algo tão trivial quanto apenas adicionar detalhes. As regras do jogo são bem diferentes, então a semelhança/isomorfismo é mais coincidência do que uma regra.

Concordo com todos os teus outros pontos sobre distinção estrita entre análise/design/código/testes (já trabalhei em times com e sem ANs, com ou sem QA, sou fã de DDD e TDD e modelos executáveis). Mas mesmo que na prática nem sempre haja diferença clara, ainda vejo valor em conceitualmente entender as diferenças.

Falou,

Rafael

2011/3/31 Marcio Tierno <mtierno@rocketmail.com>

>
>
>
> Então, Rafael.
>
> Isomórfico quer dizer "mesma forma", né? I.e., são iguais.
>
> De novo recorrendo a Jacobson, o ponto é o seguinte: você consegue pensar
> só no problema mesmo? Quando se faz uma abstração da realidade para se
> construir um modelo, já se está pensando na solução, pois a decisão sobre
> quais elementos de detalhe esconder (princípio de abstração) direciona
> decisivamente como será a construção final.
>
> Na maioria das vezes, o trabalho de modelagem situa-se numa zona cinzenta
> que pode ser melhor descrita como sendo design, mas com níveis de abstração
> variáveis.
>
> Isso para não dizer que, com o advento das abordagens ágeis, essa separação
> toda de AN/AS/D/C/T e etc deixou de fazer sentido. Que dirá separar os
> modelos desse jeito.
>
> MT.
>
> --- Em UML-BR@yahoogrupos.com.br, Rafael Chaves <rchaves.jug@...> escreveu
[Quoted text hidden]
[Quoted text hidden]
[Quoted text hidden]

---

**Charles Abreu** <charlesabreu@gmail.com> Thu, Mar 31, 2011 at 8:15 PM
Reply-To: UML-BR@yahoogrupos.com.br

To: UML-BR@yahoogrupos.com.br

Fabrício,

Não tenho certeza se entendi completamente a sua questão, mas vou tentar responder.

Acho improdutivo pensar em limitações de navegação entre classes quando estou modelando um domínio. Se uma coisa se liga à outra no mundo real, ligo as duas no modelo (1). Não vejo nenhum ganho em colocar restrições de navegação nos meus modelos de classes independentes de tecnologia. Essas restrições são desnecessárias para modelar o problema, pelo menos para mim.

Eventualmente, esses modelos de negócio (ou de domínio) - que podem existir num guardanapo, numa ferramenta ou na cabeça do programador - precisam ser mapeados para alguma tecnologia de implementação executável (2). Nesse momento entram em cena restrições ortogonais ao domínio original (problema), que são as restrições da solução tecnológica. Uma determinada tecnologia pode tornar cara a navegação em ambos os sentidos, o que me obriga, *nesse momento*, a colocar uma restrição de navegação na solução. Veja, a restrição foi colocada por limitação da solução; ela não existia no modelo original do negócio. Portanto, as associações bidirecionais não são uma limitação, mas muito pelo contrário, as unidirecionais é que são uma versão restrita do modelo original.

(1) Se não me engano, associações bidirecionais são o padrão na UML (a reta sem setas em nenhum dos lados). As associações unidirecionais são um tipo especial de associação chamadas "directed associations" (contém uma seta em uma das pontas).
(2) Isso não impede que o modelo inicial também seja executável; só uma curiosidade, irrelevante para esta discussão.

2011/3/31 Fabrício Cabral <fabriciofx@gmail.com>

> Olá Charles,
>
> primeiramente, gostaria de lhe agradecer pelo seu e-mail. O mesmo é muito
> conciso
> e detalhista. A parte do "fraco acoplamento *inter-domínios* e forte coesão
> *intra-domínio*"
> ,sem dúvida, é primoroso! :)
>
> É esse tipo de discussão, na minha opinião, que faz enriquecer essa lista!
> Parabéns a
> todos os participantes! :)
>
> Agora Charles, me permita fazer uma observação: pelo que eu entendi, você
> recomenda
> sempre usar os relacionamentos bidirecionais e utilizar o Hibernate com
> lazy
> em todas
> associações, correto? Mas a utilização do Hibernate com lazy não seria uma
> forma
> de utilizar uma tecnologia de ORM para sanar uma deficiência do modelo
> analisado?
> A deficiência no caso, seria o não mapeamento correto das associações que
> deveriam
> e as que não devam ser bidirecionais.
>
> Atenciosamente,
>
> 2011/3/30 Charles Abreu <charlesabreu@gmail.com>
>

> > Oi, Fabrício.
> >
> > Deixe-me tentar dar minha contribuição prática.
> >
> > Do ponto de vista de análise, a minha recomendação é *sempre usar
> > relacionamentos bidirecionais*. No mundo real uma coisa se liga à outra e
> > vice-versa, não é? Porque não seria igual em um modelo do mundo real? É
> > simples: pense em todas as associações como bidirecionais. Isso torna o
> > modelo muito mais simples de se lidar, e as regras e as restrições se
> tornam
> > evidentes e intuitivas.
> >
> > Na implementação, vai depender da tecnologia usada. Para quem usa o
> > Hibernate, a minha recomendação é *sempre manter todas as associações
> > bidirecionais e lazy*. Sempre. Essa é a graça do Hibernate: ele te

[Quoted text hidden]

> > original da seguinte forma: "fraco acoplamento *inter-domínios* e forte
> > coesão *intra-domínio*". E, assim, nossos relacionamentos bidirecionais
> > intra-domínio não estão mais violando nenhuma boa prática.
> >
> > A segunda questão é quanto à performance da implementação. Repetindo,
> isso
> > depende da tecnologia usada. No caso do Hibernate, não há problema,
> *desde
> > que todas as associações sejam lazy*. Nos pontos em que precisar carregar
> > os relacionamentos em uma tacada só, simplesmente adicione um "fetch
> join"
> > na query. Assim estão em produção vários sistemas, alguns com volume de
> > acessos e dados de causar espanto em muita gente experiente. Para outras
> > tecnologias, é preciso avaliar.
> >
> > Espero ter ajudado mais do que atrapalhado.
> >
> > Charles Abreu
> > http://twitter.com/#!/charlesabreu
> > <http://twitter.com/#!/charlesabreu>https://pagseguro.uol.com.br
> > <https://pagseguro.uol.com.br/>http://www.vousossegado.com.br
> >
> > 2011/3/29 Fabrício Cabral <fabriciofx@gmail.com>

[Quoted text hidden]

> ------------------------------------
>
> Links do Yahoo! Grupos
>
>
>

[As partes desta mensagem que não continham texto foram removidas]

**Marcelo Elias Del Valle** <mvallebr@gmail.com> Thu, Mar 31, 2011 at 9:23 PM
Reply-To: UML-BR@yahoogrupos.com.br
To: UML-BR@yahoogrupos.com.br
Cc: Marcio Tierno <mtierno@rocketmail.com>

Quero comentar um ponto:

"*A bidirecionalidade só é necessária por questões de implementação, nunca de projeto ou análise. Se o desenvolvedor conseguir responder aos requisitos do sistema com unidirecionalidade, deve fazê-lo. Bidirecionalidade não é requisito, é ferramenta de construção.*"

Eu concordo com isso. Contudo, para não causar confusão na cabeça do Fabrício, quero chamar atenção para um ponto: perceba que, embora a bidirecionalidade seja necessária somente por questões de implementação, ela é necessária na implementação por conta do requisito.
Não é necessário tentar achar uma maneira sistemática de, a partir do requisito diretamente, definir se a bidirecionalidade é necessária ou não, mas se na implementação você não precisar da associação, é porque o requisito não exigia. O que determina a relação é o negócio. Método para descobrir? Tente implementar sem utilizar e descubra se era necessário ou não.

MT,

Concorda?

[]s
[Quoted text hidden]
> --- Em UML-BR@yahoogrupos.com.br, Fabrício Cabral <fabriciofx@...>
[Quoted text hidden]
--
Marcelo Elias Del Valle
http://SorocabaMundi.com <http://sorocabamundi.com/> - Sorocaba Mundi - Tudo relacionado a Sorocaba
http://www.humansoftware.com.br

[As partes desta mensagem que não continham texto foram removidas]

---

Links do Yahoo! Grupos

[Quoted text hidden]

---

**Rodolpho Ugolini** <rodolpho.ugolini@gmail.com> Fri, Apr 1, 2011 at 11:31 AM
Reply-To: UML-BR@yahoogrupos.com.br
To: UML-BR@yahoogrupos.com.br
Cc: RUTHEMBERG JÚNIOR <ruthemberg@bol.com.br>

Pessoal, que discussão bacana (e pensar que começou com o Fabrício se desculpando e achando que era OFF...)

É por estas e por outras que gosto tanto desta lista: são poucas, mas excelentes as discussões. E acaloradas também!

Bom, vamos lá... fiquei 2 dias sem conseguir participar:

Ruthemberg, você disse "não existe Use Case sem UML" para poder justificar o uso da frase "deve-se modelar o "O QUE", não o "COMO". Mas caso de uso existe sem UML sim. E na verdade o diagrama de casos de uso é totalmente

opcional, já que toda a informação pode estar no documento de descrição, a sua afirmação perde o sentido. Por isto é importante deixar certas coisas claras. Não acho que é para dizer que "eu sei mais". Neste caso tem certo e errado. Afirmar que caso de uso não vive sem a UML é errado. Posso gerenciar requisitos com casos de uso, sem UML. Mas isto é papo para outra thread.

Rafael et.al (que mencionaram modelo de análise) perfeito: se considerarmos um modelo de análise, podemos usar Associações Bidirecionais. Aliás, como no modelo de Análise vale tudo (costumo dizer que é um modelo idealizado, onde não existem restrições de tempo, espaço ou recursos) é até recomendado.

Mas quando partimos "prás cabeças" reforço a recomendação: sejam econômicos. Enxuguem o modelo. Desacoplem as classes o máximo possível. No longo prazo vocês vão agradecer. Como o MT disse: na codificação, sempre é mais trabalhoso criar as associações bidirecionais. Isto precisa ser implementado explicitamente "nas duas pontas".

Pergunta: o que se ganha ao criar associações bidirecionais desnecessariamente? Sendo que caso torne-se necessária é só adicionar algumas linhas de código numa das pontas depois?

Outras: você prefere gastar mais recursos de máquina? Prefere gastar mais tempo codificando? Prefere correr o risco de criar um "tungle" (nó) no código? Prefere dificultar uma refatorarção do código por conta de um acoplamento maior? Prefere ter de confiar sempre em um artifício (Lazy Load) do que num bom design?

Alias, deixar todo o trabalho para um mecanismo de Lazy Load é como dirigir como um louco só porque o carro possui ABS, Airbag e sistema anticolisão.

Para finalizar, não consegui deixar de pensar nesta frase que li: "Gargalos, concorencias, etc... cabem a profissionais de Infra que devem saber o que é preciso para atender o sistema modelado". Isso me lembra de uma seguradora que visitei a algumas semanas e que está com um problema MUITO sério por conta da arquitetura definida, que é linda no papel, mas que impede que o pessoal de infra utilize recursos de load balancing e alta disponibilidade... justamente por problemas de alto acoplamento!!!

Gargalos, concorrencias, etc são sim problemas EQUIPE DE DESENVOLVIMENTO! Aliás, concorrencia, balanceamento, disponibilidade, tudo isto pode ser resolvido com um bom design.

Por isto, vos digo: sempre que possível, no **design/implementação**, evitem bidirecionalidade entre elementos (classes, pacotes, subsistemas, etc). Já em análise, vale tudo.

GRAAANDE ABRAÇO A TODOS!

2011/3/31 RUTHEMBERG JÚNIOR <ruthemberg@bol.com.br>

> 
> 
> 
> Gente, vamos parar de procurar pelo em ovo.
> Vc vive na época do velho Jaco? A propósito, o documento Caso de Uso como
> conhecemos hoje, foi aprimorada e proposta por Alistair Cockburn.
> Poxa, além do mais, no que esse tipo de "eu sei mais do que você" contribui
> para questão levantada?
> Esse tempo dispensado para ficar batendo de frente (sem méritos algum),
> seria com certeza muito melhor aproveitado pela lista se empregado na
> resposta e auxílio aos membros do forum.
> 
> 
> []´s
> 

> RUTHEMBERG JÚNIOR
> Consultor de TI
> Analista de Sistemas
> cel 19 97136252
>
> Em 31/03/2011 14:43, Marcio Tierno < mtierno@rocketmail.com > escreveu:
>
>
>
> Ruthemberg
>
> Caso de uso EXISTE sim sem UML. Inclusive, Casos de Uso foram criados pelo
> vovô Jacobson VÁÁÁRIOS anos antes da UML existir.
>
> MT.
>
> --- Em UML-BR@yahoogrupos.com.br, RUTHEMBERG JÚNIOR escreveu
[Quoted text hidden]
[Quoted text hidden]

---

**Rodolpho Ugolini** <rodolpho.ugolini@gmail.com> Fri, Apr 1, 2011 at 11:34 AM
Reply-To: UML-BR@yahoogrupos.com.br
To: UML-BR@yahoogrupos.com.br
Cc: RUTHEMBERG JÚNIOR <ruthemberg@bol.com.br>

E à propósito: é mais fácil velejar contra o vento que contra um relacionamento unidirecional.

:-)))
[Quoted text hidden]

---

**Fabrício Cabral** <fabriciofx@gmail.com> Mon, Apr 4, 2011 at 3:18 AM
To: UML-BR@yahoogrupos.com.br

Olá Marcio,

> 1) O ponto em questão é: existe alguma maneira *sistemática* de identificar que uma associação não precise (ou precise) ser bidirecional?
>
> R: Tem. Use unidirecional por definição. Se precisar ser bidirecional, essa necessidade se fará explícita e gritante em algum ponto na codificação. Um refactoring básico e tá tudo resolvido.

Certo. O que eu estava pensando é se haveria alguma forma de se evitar este refactoring, como por exemplo a clássica pergunta para se saber se um relacionamento entre duas classes deverá ser uma herança ou uma agregação: "a classe A é uma ou tem uma classe B?" Até onde eu sei, se a resposta fosse "é uma" as chances deste relacionamento ser uma herança é bem maior. Achei que houvesse alguma coisa do gênero para saber se o relacionamento deveria ser bidirecional ou unidirecional.

> 2) Aqui eu iria analisar com cuidado. E se o programador *achar* que a bidirecionalidade não era necessária e implementar uma unidirecionalidade quando não o deveria fazer? Afinal, acredito que era papel do analista analisar o problema por todos os lados e dizer: "olha aqui realmente deveria haver uma associação bidirecional pela razão X e Y".
>
> A bidirecionalidade só é necessária por questões de implementação, nunca de projeto ou análise. Se o desenvolvedor conseguir responder aos requisitos do sistema com unidirecionalidade, deve fazê-lo. Bidirecionalidade não é requisito, é ferramenta de construção.

Marcio, quer dizer em um projeto ou análise haveria a necessidade de uma bidirecionalidade? Porque eu entendo a bidirecionalidade como uma "obrigação" que as duas classes relacionadas devem se conhecer um a outra. Por exemplo, imagine que você estivesse modelando um sistema com duas

classes, uma chamada Cérebro e outra chamada Corpo. Em sua análise você descobriu que:

1) Todo Corpo possui um único Cérebro e todo Cérebro possui um único corpo;

2) Um Cérebro não vive sem um Corpo e o Corpo não vive sem o Cérebro;

Assim, por estas razões acredito que deva haver uma bidirecionalidade entre estas classes, pois se você criar uma instância de Cérebro ela deve conter uma referência a qual Corpo ele está associado e se você criar um Corpo, ele deve conter uma referência sobre qual Cérebro ele está associado.

Agora imagine que a troca de cérebros entre corpos seja possível (isto me lembrou um desenho do Pica-Pau): Então por (1) e por (2) o relacionamento bidirecional deveria ser atualizado.

Se meu raciocínio estiver correto, então é possível na fase de projeto ou de análise, descobrir se o relacionamento entre duas classes deva ser bidirecional, certo? Agora a grande pergunta é: meu raciocínio está correto? Se não estiver, onde está a falha?

> 3) > Não é possível usar Lazy Load em uma associação de um para um?
>
> R: Possível é, só não faz sentido. Dê uma revisada no pattern e chegue à conclusão. Construir todo um aparato para carregar objetos conforme a necessidade e descobrir que há um único objeto a carregar - e que normalmente já estará carregado...

É realmente é muito óbvio. Não sei como eu fiz essa pergunta boba... :)

> 4) obviamente eu faria e refaria testes de escalabilidade antes de colocar o sistema em produção, certo? ;)
>
> Certo. A diferença seria a quantidade de mexidas a serem feitas no sistema...

Ok. :)

> 5) eu não seu como navegar na "direção contrária" de uma associação unidirecional. Seria utilizando recursos de reflection?
>
> Tem vários jeitos. Reflection é um jeito bem sofisticado. Alguns usam apoio do banco de dados. Outros "passeiam" pelos objetos até achar seu "dono". Se a navegação no sentido contrário é muito frequente e ela começa a comer recursos importantes do sistema, eis aí a necessidade da bidirecionalidade.

Desculpe a pergunta boba, mas como seria este "passear entre os objetos até achar o seu dono"? Os objetos seriam que estar agrupados em uma coleção e eu sair verificando um a um até achar uma referência do objeto a qual eu esteja interessado (dono)?

> Muito boas as suas perguntas e comentários. Parabéns!!!!!

Ah, obrigado! E eu agradeço também por sua gentileza em respondê-las. Acho muita generosidade não só sua, mas de todos da lista, de contribuir com a sua experiência e conhecimento. :)

Atenciosamente,

--
--fx

---

**Fabrício Cabral** <fabriciofx@gmail.com> Mon, Apr 4, 2011 at 3:35 AM
To: UML-BR@yahoogrupos.com.br

Olá Charles,

2011/3/31 Charles Abreu <charlesabreu@gmail.com>

> Fabrício,
>
> Não tenho certeza se entendi completamente a sua questão, mas vou tentar responder.

Obrigado por tentar me ajudar... :)

> Acho improdutivo pensar em limitações de navegação entre classes quando estou modelando um domínio. Se uma coisa se liga à outra no mundo real, ligo as duas no modelo (1). Não vejo nenhum ganho em colocar restrições de navegação nos meus modelos de classes independentes de tecnologia. Essas restrições são desnecessárias para modelar o problema, pelo menos para mim.

Eu não sei estou falando besteira, mas até onde eu posso ver, quando duas classes A e B estão associadas, elas "se conhecem uma a outra". Mas parece-me que quando você mantém a bidirecionalidade (ou limita a associação a ser unidirecional) é como se você quisesse dizer mais coisas a esta associação, mas de uma maneira "implicita", digamos assim. Tome como exemplo uma associação entre duas classes A e B, onde a multiplicidade do lado B fosse 0..1. Este "0" aí na multiplicidade indica que um objeto da classe A *pode ou não* estar associado a um objeto da classe B. Isto, é, há uma opcionalidade no relacionamento. Veja (pelo menos eu entendo assim) que este entendimento é uma coisa ímplicita. Não está escrito com todas as letras no modelo esta restrição. Quem for ler  o modelo deverá ter uma certa "malícia" (digamos assim :) ) para interpretar corretamente.

> Eventualmente, esses modelos de negócio (ou de domínio) - que podem existir num guardanapo, numa ferramenta ou na cabeça do programador - precisam ser mapeados para alguma tecnologia de implementação executável (2). Nesse momento entram em cena restrições ortogonais ao domínio original (problema), que são as restrições da solução tecnológica. Uma determinada tecnologia pode tornar cara a navegação em ambos os sentidos, o que me obriga, *nesse momento*, a colocar uma restrição de navegação na solução. Veja, a restrição foi colocada por limitação da solução; ela não existia no modelo original do negócio. Portanto, as associações bidirecionais não são uma limitação, mas muito pelo contrário, as unidirecionais é que são uma versão restrita do modelo original.

É justamente este o ponto: esta restrição propiciada pela associação undirecional tem função apenas simplificar ou otimizar a implementação ou há alguma outra informação semântica (como o caso da mutiplicidade que eu citei anteriormente) implícita aí?

> (1) Se não me engano, associações bidirecionais são o padrão na UML (a reta sem setas em nenhum dos lados). As associações unidirecionais são um tipo especial de associação chamadas "directed associations" (contém uma seta em uma das pontas).
> (2) Isso não impede que o modelo inicial também seja executável; só uma curiosidade, irrelevante para esta discussão.

Atenciosamente,

--
--fx

---

**Fabrício Cabral** <fabriciofx@gmail.com> Mon, Apr 4, 2011 at 3:44 AM
To: UML-BR@yahoogrupos.com.br

Olá Marcelo,

2011/3/31 Marcelo Elias Del Valle <mvallebr@gmail.com>

> Quero comentar um ponto:
>
> "*A bidirecionalidade só é necessária por questões de implementação, nunca de projeto ou análise. Se o desenvolvedor conseguir responder aos requisitos do sistema com unidirecionalidade, deve fazê-lo. Bidirecionalidade não é requisito, é ferramenta de construção.*"
>
> Eu concordo com isso. Contudo, para não causar confusão na cabeça do Fabrício, quero chamar atenção para um ponto: perceba que, embora a bidirecionalidade seja necessária somente por questões de implementação, ela é necessária na implementação por conta do requisito.

Marcelo, eu sei que você não queria causar confusão na minha cabeça, mas foi inevitável... :)

O me chamou a atenção foi: se a bidirecionalidade é necessária apenas por uma questão de implementação e a ela é necessária na implementação por conta do requisito, o modelo UML do sistema não deveria refletir também este requisito? Trocando em miúdos: se é um requisito do negócio ou de implementação, ele não deveria estar em algum lugar no modelo?

> Não é necessário tentar achar uma maneira sistemática de, a partir do requisito diretamente, definir se a bidirecionalidade é necessária ou não, mas se na implementação você não precisar da associação, é porque o requisito não exigia. O que determina a relação é o negócio. Método para descobrir? Tente implementar sem utilizar e descubra se era necessário ou não.

Minha ideia de obter uma "maneira sistemática", digamos assim, é tentar inferir esta necessidade sem ter que descobri-la na hora da implementação. Não sei se é normal, durante a fase de implementação, você saber que os requisitos X, Y e Z do modelo deveriam ser outros. Passa a (talvez falsa) impressão que a modelagem do negócio foi feita na base do achismo.

Atenciosamente,

--
--fx

---

**Marcelo Elias Del Valle** <mvallebr@gmail.com> Mon, Apr 4, 2011 at 1:19 PM
To: UML-BR@yahoogrupos.com.br
Cc: Fabrício Cabral <fabriciofx@gmail.com>

Fabrício,

Quero comentar 2 pontos desse seu email:

"*Marcio, quer dizer em um projeto ou análise haveria a necessidade de uma bidirecionalidade? Porque eu entendo a bidirecionalidade como uma "obrigação" que as duas classes relacionadas devem se conhecer um a outra.*"

De fato, o modelo estará de acordo com a implementação caso uma conheça a outra. Não confunda isso, contudo, com ciclo de vida: você pode ter uma relação bidirecional entre duas classes e instanciar somente uma delas. Se o relacionamento for uma composição, aí sim estaremos falando de ciclo de vida.

"*É realmente é muito óbvio. Não sei como eu fiz essa pergunta boba... :)*"

Sem essa de pergunta boba. Se te xingarem por questionar, deixe que xinguem. O objetivo da lista é exatamente esse tipo de discussão, continue perguntando.

Abraços,
Marcelo.

[Quoted text hidden]

[Quoted text hidden]

--
--fx

[As partes desta mensagem que não continham texto foram removidas]



--
Marcelo Elias Del Valle
http://SorocabaMundi.com - Sorocaba Mundi - Tudo relacionado a Sorocaba
http://www.humansoftware.com.br

---

**Marcelo Elias Del Valle** <mvallebr@gmail.com> Mon, Apr 4, 2011 at 1:38 PM
To: UML-BR@yahoogrupos.com.br
Cc: Fabrício Cabral <fabriciofx@gmail.com>

Fabrício,

Estamos convergindo para um entendimento! :D Seguem as respostas inline.

> Marcelo, eu sei que você não queria causar confusão na minha cabeça, mas foi inevitável... :)
>
> O me chamou a atenção foi: se a bidirecionalidade é necessária apenas por uma questão de implementação e a ela é necessária na implementação por conta do requisito, o modelo UML do sistema não deveria refletir também este requisito? Trocando em miúdos: se é um requisito do negócio ou de implementação, ele não deveria estar em algum lugar no modelo?

A pergunta já mostra que você está entendendo o que está acontecendo. A questão é: qual a finalidade do modelo? É preciso lembrar que UML é só a linguagem... Ela não determina o que você diz. Eu posso dizer algo em português ou em inglês, mas **o que** eu estou dizendo pode ser exatamente a mesma coisa. Do mesmo modo, o fato de eu dizer algo em inglês não implica em que ter que mudar o que eu estou dizendo, eu apenas estou fazendo isso numa linguagem diferente.

Usamos uma linguagem de programação para dizer algo ao computador. A linguagem de programação necessita ser extremamente formal exatamente porque o computador é uma máquina totalmente lógica. Quando falamos com um ser humano, não somos tão formais, mas temos possibilidade de entendimento ambíguo, em contrapartida. Não há como você escrever algo para o computador e o mesmo ficar em dúvida, a linguagem de computador é uma linguagem EXATA.

UML foi especificada com bastante formalismo, de modo que pode ser possível usá-la para dizer ao computador o que fazer, como é o caso da MDA (Model Driven Architecture). Contudo, a maioria usa a linguagem para especificar coisas para serem humanos. Essa é a finalidade para a qual ela é normalmente utilizada.

Dizer que a associação é bidirecional sem que haja a necessidade de negócio pode até ser considerado inconsistente. Por que? Porque você está criando um modelo, no computador, que faz mais do que é necessário, através da geração de acoplamento. Contudo, você concorda que se colocar essa relação no modelo UML e não colocar no modelo escrito em código, isso não teve conseqüência negativa prática nenhuma?

Acho que a questão aqui é diferenciarmos a filosofia do pragmatismo. A relação não é bidirecional, ok... Mas qual método prático vamos usar para resolver o problema?

> Não é necessário tentar achar uma maneira sistemática de, a partir do
> requisito diretamente, definir se a bidirecionalidade é necessária ou não,
> mas se na implementação você não precisar da associação, é porque o
> requisito não exigia. O que determina a relação é o negócio. Método para
> descobrir? Tente implementar sem utilizar e descubra se era necessário ou
> não.
>

Minha ideia de obter uma "maneira sistemática", digamos assim, é tentar inferir esta necessidade sem ter que descobri-la na hora da implementação. Não sei se é normal, durante a fase de implementação, você saber que os requisitos X, Y e Z do modelo deveriam ser outros. Passa a (talvez falsa) impressão que a modelagem do negócio foi feita na base do achismo.

UML serve para tentar integrar o negócio com o desenvolvimento, tarefa que muitas tecnologias e métodos tentam cumprir. Se você usar UML com MDA, não pode mesmo ser feito assim, você terá de ser tão formal quanto o computador exige. Contudo, se estiver usando UML como ferramenta para dizer a um desenvolvedor o que fazer, seus diagramas por vezes "acharão" alguma coisa sobre o negócio e por vezes "acharão" alguma coisa sobre o código. Quando usado dessa segunda forma, UML é mesmo uma aproximação de ambos, para tentar juntar, o máximo que for possível, a visão do programador e do usuário. Criar algo exato necessitaria ter em UML o mesmo formalismo que você tem numa linguagem de programação e, se estiver adotando esse approach, estará indo na linha do MDA.

Respondi?

Abraços,
--
Marcelo Elias Del Valle
http://SorocabaMundi.com - Sorocaba Mundi - Tudo relacionado a Sorocaba
http://www.humansoftware.com.br

---

**Marcio Tierno** <mtierno@rocketmail.com> Mon, Apr 4, 2011 at 9:01 PM
Reply-To: UML-BR@yahoogrupos.com.br
To: UML-BR@yahoogrupos.com.br

Fabricio

1) Como eu disse, a maneira sistemática é: use unidirecionalidade por defninição e bidirecionalidade se absolutamente necessário.

2) Vc está usando uma abordagem ontológica, i.e., baseada no que o objeto "é". Num projeto de software, o ideal é usar uma abordagem funcional, i.e., baseada no que um objeto "faz". Assim, no seu exemplo, não importa muito o que um cérebro ou um corpo são um em relação ao outro mas, sim, o que fazem. O Cérebro precisará "ver" o corpo se precisar enviar-lhe uma mensagem e vice-versa. Aparentemente, trata-se de bidirecionalidade, certo? Bem, um mesmo conjunto de classes terá comportamento diferente para sistemas diferentes. Então, se você tem um sistema no qual vai-se do cérebro para o corpo frequentemente mas só se vai do corpo para o cérebro muito de vez em quando, então há aí uma oportunidade de unidirecionalidade (i.e., economia de design e de recursos computacionais) a ser explorada.

Outro exemplo: relação pai e filho. Um pai tem filhos e um filho obrigatoriamente tem um pai, certo? Então, é bidirecional.

Bom, num sistema de seguro de vida, no qual o filho é o beneficiário, é unidirecional de pai para filho, pois nunca se precisa fazer a navegação em contrário.

Já num sistema de seguro saúde, aí vc pode querer saber quem é o pai a partir do filho, caso este último esteja em atendimento de emergência num hospital e se necessite contatar o titular do plano. Já o contrário, de pai para filho, por incrível que pareceça, será bem pouco necessário nesse sistema e a navegabilidade poderá ser "ao contrário", i.e., de filho para pai mas não de pai para filho. "Lôco", né?

5) É isso aí. Por exemplo.

[]s!!!

MT.

--- Em UML-BR@yahoogrupos.com.br, Fabrício Cabral <fabriciofx@...> escreveu

[Quoted text hidden]
> [As partes desta mensagem que não continham texto foram removidas]
>



---

**Marcio Tierno** <mtierno@rocketmail.com> Mon, Apr 4, 2011 at 9:06 PM
Reply-To: UML-BR@yahoogrupos.com.br
To: UML-BR@yahoogrupos.com.br

Srs

Bidirecionalidade é um mecanismo de construção e JAMAIS será requisito de negócio, da mesma forma que o uso de tabelas num banco relacional - necessárias para implementar as regras de negócio, mas não parte delas.

O negócio pode exigir que, por exemplo, dado um conjunto de pais se obtenha a soma das idades de seus filhos; ou dado um conjunto de filhos, que se saiba qual porcentagem de pais é palmeirense ou santista. A maneira mais fácil de implementar isso talves seja por bidirecionalidade. OU então mandar às favas o modelo de objetos e conseguir o feito com uma query SQL de duas linhas, sei lá. Mas nem uma (a bidirecionalidade) nem a outra (a query) serão requisitos de negócio, apenas mecanismo de construção para implementar os requisitos.

[]s!!!

MT.

--- Em UML-BR@yahoogrupos.com.br, Marcelo Elias Del Valle <mvallebr@...> escreveu
>
> Fabrício,
>
> Estamos convergindo para um entendimento! :D Seguem as respostas inline.
>
> Marcelo, eu sei que você não queria causar confusão na minha cabeça, mas
> > foi inevitável... :)
> >
> > O me chamou a atenção foi: se a bidirecionalidade é necessária apenas por
> > uma questão de implementação e a ela é necessária na implementação por
> > conta do requisito, o modelo UML do sistema não deveria refletir também
> > este requisito? Trocando em miúdos: se é um requisito do negócio ou de
> > implementação, ele não deveria estar em algum lugar no modelo?
> >
>
> A pergunta já mostra que você está entendendo o que está acontecendo. A
> questão é: qual a finalidade do modelo? É preciso lembrar que UML é só a
> linguagem... Ela não determina o que você diz. Eu posso dizer algo em
> português ou em inglês, mas *o que* eu estou dizendo pode ser exatamente a

[Quoted text hidden]
> http://SorocabaMundi.com <http://sorocabamundi.com/> - Sorocaba Mundi - Tudo
> relacionado a Sorocaba
> http://www.humansoftware.com.br
>
>
> [As partes desta mensagem que não continham texto foram removidas]
>



---

**Marcelo Elias Del Valle** <mvallebr@gmail.com> Tue, Apr 5, 2011 at 12:09 AM
Reply-To: UML-BR@yahoogrupos.com.br
To: UML-BR@yahoogrupos.com.br
Cc: Marcio Tierno <mtierno@rocketmail.com>

MT,

Veja bem, uma coisa é não ser requisito de negócio, o que realmente não é. Outra coisa é ser consequência do requisito de negócio.
É o requisito que determina se há ou não a bidirecionalidade, pois dado o requisito, a birecionalidade pode vir a ser necessária ou não, concorda?
Para que colocar uma relação bidirecional se o negócio não pede? Se o negócio não pede, é melhor não colocar na implementação. Se é melhor não colocar na implementação, deve ir no modelo? Na minha opinião, depende do formalismo do modelo.

Abraços,
Marcelo.

Em 4 de abril de 2011 21:06, Marcio Tierno <mtierno@rocketmail.com>escreveu:

> Srs
> Bidirecionalidade é um mecanismo de construção e JAMAIS será requisito de
> negócio, da mesma forma que o uso de tabelas num banco relacional -
> necessárias para implementar as regras de negócio, mas não parte delas.
> O negócio pode exigir que, por exemplo, dado um conjunto de pais se obtenha
> a soma das idades de seus filhos; ou dado um conjunto de filhos, que se
> saiba qual porcentagem de pais é palmeirense ou santista. A maneira mais
> fácil de implementar isso talves seja por bidirecionalidade. OU então mandar
> às favas o modelo de objetos e conseguir o feito com uma query SQL de duas
> linhas, sei lá. Mas nem uma (a bidirecionalidade) nem a outra (a query)
> serão requisitos de negócio, apenas mecanismo de construção para implementar
> os requisitos.
> []s!!!
> MT.
> --- Em UML-BR@yahoogrupos.com.br, Marcelo Elias Del Valle <mvallebr@...>
[Quoted text hidden]
> > http://SorocabaMundi.com <http://sorocabamundi.com/> - Sorocaba Mundi -
> Tudo
> > relacionado a Sorocaba
> > http://www.humansoftware.com.br
> [As partes desta mensagem que não continham texto foram removidas]

--
Marcelo Elias Del Valle
http://SorocabaMundi.com <http://sorocabamundi.com/> - Sorocaba Mundi - Tudo
relacionado a Sorocaba
http://www.humansoftware.com.br

[As partes desta mensagem que não continham texto foram removidas]

[Quoted text hidden]

---

**Marcelo Andrade** <mfandrade@gmail.com> Tue, Apr 5, 2011 at 12:54 PM
Reply-To: UML-BR@yahoogrupos.com.br
To: UML-BR@yahoogrupos.com.br

2011/4/4 Marcio Tierno <mtierno@rocketmail.com>:
> Srs

> Bidirecionalidade é um mecanismo de construção e JAMAIS será requisito de negócio, da mesma forma que o uso de tabelas num banco relacional - necessárias para implementar as regras de negócio, mas não parte delas.

Nossa! Ficou extensa a discussão e pode ser que tenha perdido algo.

Mas eu concordo com este ponto de vista do MT e vou além. Entendo que na modelagem OO devemos tentar refletir o mundo real. E, a menos a meu ver, a maioria das associações entre objetos no mundo real é unidirecional.

Talvez seja fácil ilustrar isso em exemplos clássicos como do automóvel. No meu entendimento, um dado automóvel (i.e., uma dada instância de automóvel) "conhece" suas peças. Mas uma dada peça não precisa conhecer o automóvel (sei lá... não se precisa gravar o número do chassi nos pneus, no volante, na repimboca da parafuseta), por isso mesmo as peças são intercambiáveis. Da mesma forma, o dono conhece seu automóvel, mas o automóvel não precisa conhecer o dono (a menos que você personalize seu automóvel de tal forma que só você consiga "mandar mensagens" para dirigir o automóvel, p.ex.). E por aí vai...

Claro que isso depende da visão de mundo de cada um. Chutando um modelo, p.ex., entre páginas e livros, num dado contexto eu poderia dizer que o livro conhece suas páginas e as página também precisa conhecer o livro (para auxiliar na montagem do sumário, ou o que seja). Não quer dizer que é a verdade, podem ser apenas visões de mundo distintas.

Além disso, as discussões sobre impacto das associações bidirecionais que, em geral, são mais custosas em termos de implementação. O que corrobora meu entendimento de que não porque usá-las indiscriminadamente.

O que nos leva ao ponto onde eu queria chegar: será que o fato de queremos usar associações bidirecionais tão indiscriminadamente não evidencia que nossos modelos OO estão falhando em seu propósito de universalizar a comunicação? (P.ex., para nosso sistema se eu acho que tal associação seja unidirecional e você, bidirecional, colocamos bidirecional, não discutimos, e confiamos nas ferramentas para arcar com os impactos de implementação).

Ou noutras palavras, será que estamos realmente sabendo modelar OO?

Atts.

--
MARCELO F ANDRADE
Belem, Amazonia, Brazil

"I took the red pill"

—,—.—,—

**Marcelo Elias Del Valle** <mvallebr@gmail.com> Tue, Apr 5, 2011 at 1:37 PM
Reply-To: UML-BR@yahoogrupos.com.br
To: UML-BR@yahoogrupos.com.br

Marcelo Andrade,

Eu pensei que tinha entendido o que você e o MT estavam pensando sobre a modelagem, mas agora estou vendo que realmente discordamos.
O que você chama de "mundo real", pra mim é o requisito. Não existe "mundo real". Existe requisito. Se o requisito estiver detalhando um mundo da fantasia, é o mundo da fantasia que vale.
O modelo não tem que estar de acordo com o mundo real, nem o código, nem todo o resto da implementação. A implementação, modelo, etc, tem que estar de acordo com o requisito.
Eu aceito que vocês digam que o requisito não tem que definir nada a respeito do modelo, como não tem mesmo. Não tem que estar descrito no caso de uso ou no user story que a relação entre entidade A e B deve ser de jeito A ou jeito B, isso é detalhe de implementação. Mas a implementação deve ser feita de acordo com o requisito e não de acordo com o mundo real.
Exemplo: peça e automóvel. Não importa como peça e automóvel estão relacionados no mundo real, peça vai ter associação com automóvel se você precisar disso na implementação, por conta do requisito. Se no requisito consta que você precisa listar as peças de um automóvel e em outra tela você precisa listar os automóveis que usam determinada peça, na sua implementação você vai sentir necessidade da bidirecionalidade. Se só existir o requisito de listar as peças de um automóvel, você só sentirá falta da associação unidirecional.

Abraços,
Marcelo Valle.

Em 5 de abril de 2011 12:54, Marcelo Andrade <mfandrade@gmail.com> escreveu:
[Quoted text hidden]
--
Marcelo Elias Del Valle
http://SorocabaMundi.com <http://sorocabamundi.com/> - Sorocaba Mundi - Tudo relacionado a Sorocaba
http://www.humansoftware.com.br

[As partes desta mensagem que não continham texto foram removidas]

[Quoted text hidden]

**Marcio Tierno** <mtierno@rocketmail.com> Wed, Apr 6, 2011 at 11:26 AM
Reply-To: UML-BR@yahoogrupos.com.br
To: UML-BR@yahoogrupos.com.br

Marcelo

Na verdade, vc continua concordando...rssss!!

No seu exemplo, em nenhum momento o requisito declara que precisa de uma bidirecionalidade. Vc, com sua experiencia, acredita que essa bidirecionalidade sera' necessaria para atender ao requisito, mas ela nao e' obrigatoria.

A prova de que a bidirecionalidade nao esta nem implicita nem explicitamente expressa no requisito e' a seguinte: sera' que eu consigo implementar esse requisito numa linguagem procedural nao orientada a objeto? Se a resposta for sim (e e'!!!), entao nao ha bidirecionalidade expressa no requisito - pois, se houvesse, so' seria possivel desenvolver o sistema numa linguagem OO.

CDQ, certo?

MT.

--- Em UML-BR@yahoogrupos.com.br, Marcelo Elias Del Valle <mvallebr@...> escreveu
>
> Marcelo Andrade,
>
> Eu pensei que tinha entendido o que você e o MT estavam pensando sobre
> a modelagem, mas agora estou vendo que realmente discordamos.
> O que você chama de "mundo real", pra mim é o requisito. Não existe
> "mundo real". Existe requisito. Se o requisito estiver detalhando um mundo
> da fantasia, é o mundo da fantasia que vale.
> O modelo não tem que estar de acordo com o mundo real, nem o código,
> nem todo o resto da implementação. A implementação, modelo, etc, tem que
> estar de acordo com o requisito.
> Eu aceito que vocês digam que o requisito não tem que definir nada a
> respeito do modelo, como não tem mesmo. Não tem que estar descrito no caso
> de uso ou no user story que a relação entre entidade A e B deve ser de jeito
> A ou jeito B, isso é detalhe de implementação. Mas a implementação deve ser
> feita de acordo com o requisito e não de acordo com o mundo real.
> Exemplo: peça e automóvel. Não importa como peça e automóvel estão
> relacionados no mundo real, peça vai ter associação com automóvel se você
> precisar disso na implementação, por conta do requisito. Se no requisito
> consta que você precisa listar as peças de um automóvel e em outra tela você
> precisa listar os automóveis que usam determinada peça, na sua implementação
> você vai sentir necessidade da bidirecionalidade. Se só existir o requisito
> de listar as peças de um automóvel, você só sentirá falta da associação
> unidirecional.
>
> Abraços,
> Marcelo Valle.
>
> Em 5 de abril de 2011 12:54, Marcelo Andrade <mfandrade@...> escreveu:
>
> >
> >
> > 2011/4/4 Marcio Tierno <mtierno@...>:

[Quoted text hidden]
> http://SorocabaMundi.com <http://sorocabamundi.com/> - Sorocaba Mundi - Tudo
> relacionado a Sorocaba
> http://www.humansoftware.com.br
>
>
> [As partes desta mensagem que não continham texto foram removidas]
>

__,_._,___

---

**Marcelo Elias Del Valle** <mvallebr@gmail.com> Wed, Apr 6, 2011 at 12:45 PM
Reply-To: UML-BR@yahoogrupos.com.br
To: UML-BR@yahoogrupos.com.br
Cc: Marcio Tierno <mtierno@rocketmail.com>

MT,

Certo, é isso aí mesmo! Embora a implementação seja uma consequência dos requisitos, a decisão de haver bidirecionalidade ou não é uma decisão de implementação.
Diga-se de passagem, é muito ruim quando recebemos requisitos tentando impor uma forma de implementar, como por vezes acontece. O correto é deixar essa decisão para o pessoal técnico, ou seja, que programa.

Abraços,
Marcelo.

Em 6 de abril de 2011 11:26, Marcio Tierno <mtierno@rocketmail.com>escreveu:

> 
> 
> Marcelo
> 
> Na verdade, vc continua concordando...rssss!!
> 
> No seu exemplo, em nenhum momento o requisito declara que precisa de uma
> bidirecionalidade. Vc, com sua experiencia, acredita que essa
> bidirecionalidade sera' necessaria para atender ao requisito, mas ela nao e'
> obrigatoria.
> 
> A prova de que a bidirecionalidade nao esta nem implicita nem
> explicitamente expressa no requisito e' a seguinte: sera' que eu consigo
> implementar esse requisito numa linguagem procedural nao orientada a objeto?
> Se a resposta for sim (e e'!!!), entao nao ha bidirecionalidade expressa no
> requisito - pois, se houvesse, so' seria possivel desenvolver o sistema numa
> linguagem OO.
> 
> CDQ, certo?
> 
> 
> MT.
> 
> --- Em UML-BR@yahoogrupos.com.br, Marcelo Elias Del Valle <mvallebr@...>
[Quoted text hidden]
> > http://SorocabaMundi.com <http://sorocabamundi.com/> - Sorocaba Mundi -
> Tudo
> 
> > relacionado a Sorocaba
> > http://www.humansoftware.com.br
> >
> >
> > [As partes desta mensagem que não continham texto foram removidas]
> >
> 
> 
> 
> 

--
Marcelo Elias Del Valle

http://SorocabaMundi.com <http://sorocabamundi.com/> - Sorocaba Mundi - Tudo relacionado a Sorocaba
http://www.humansoftware.com.br

[As partes desta mensagem que não continham texto foram removidas]

[Quoted text hidden]

---

**Fabrício Cabral** <fabriciofx@gmail.com> Mon, Apr 11, 2011 at 9:41 PM
To: UML-BR@yahoogrupos.com.br

Olá Marcelo,

desculpe a demora em responder o seu e-mail. Muitas coisas a fazer... :)

2011/4/4 Marcelo Elias Del Valle <mvallebr@gmail.com>

> A pergunta já mostra que você está entendendo o que está acontecendo. A questão é: qual a finalidade do modelo? É preciso lembrar que UML é só a linguagem... Ela não determina o que você diz. Eu posso dizer algo em português ou em inglês, mas **o que** eu estou dizendo pode ser exatamente a mesma coisa. Do mesmo modo, o fato de eu dizer algo em inglês não implica em que ter que mudar o que eu estou dizendo, eu apenas estou fazendo isso numa linguagem diferente.
> Usamos uma linguagem de programação para dizer algo ao computador. A linguagem de programação necessita ser extremamente formal exatamente porque o computador é uma máquina totalmente lógica. Quando falamos com um ser humano, não somos tão formais, mas temos possibilidade de entendimento ambíguo, em contrapartida. Não há como você escrever algo para o computador e o mesmo ficar em dúvida, a linguagem de computador é uma linguagem EXATA.
> UML foi especificada com bastante formalismo, de modo que pode ser possível usá-la para dizer ao computador o que fazer, como é o caso da MDA (Model Driven Architecture). Contudo, a maioria usa a linguagem para especificar coisas para serem humanos. Essa é a finalidade para a qual ela é normalmente utilizada.

Eu não estudei a UML a fundo, mas pelo que eu vi até agora, deu pra ver que ela realmente é baseada em um forte formalismo.

Sobre MDA eu não tenho muito o que dizer. Já ouvi falar dela, mas nunca a estudei nem fui atrás de saber seu funcionamento. Mas não é coisa de outro mundo achar que dado um modelo bem feito em UML, gerar um código muito próximo do seu sistema. :)

> Dizer que a associação é bidirecional sem que haja a necessidade de negócio pode até ser considerado inconsistente. Por que? Porque você está criando um modelo, no computador, que faz mais do que é necessário, através da geração de acoplamento. Contudo, você concorda que se colocar essa relação no modelo UML e não colocar no modelo escrito em código, isso não teve conseqüência negativa prática nenhuma?
> Acho que a questão aqui é diferenciarmos a filosofia do pragmatismo. A relação não é bidirecional, ok... Mas qual método prático vamos usar para resolver o problema?

Realmente, se a bidirecionalidade estiver no modelo e não estiver no código, não há efeito prático algum. Só a documentação do mesmo é que estaria desatualizada.

Quanto a filososia e o pragmatismo, o meu esforço nesta questão toda de bidirecionalidade é *realmente* entender o problema e saber que por que aquela bidirecionalidade está lá no modelo e o que as técnicas de modelagem me asseguram que ela realmente deva estar lá.

> UML serve para tentar integrar o negócio com o desenvolvimento, tarefa que muitas tecnologias e métodos tentam cumprir. Se você usar UML com MDA, não pode mesmo ser feito assim, você terá de ser tão formal quanto o computador exige. Contudo, se estiver usando UML como ferramenta para dizer a um desenvolvedor o que fazer, seus diagramas por vezes "acharão" alguma coisa sobre o negócio e por vezes "acharão" alguma coisa sobre o código. Quando usado dessa segunda forma, UML é mesmo uma

> aproximação de ambos, para tentar juntar, o máximo que for possível, a visão do programador e do usuário. Criar algo exato necessitaria ter em UML o mesmo formalismo que você tem numa linguagem de programação e, se estiver adotando esse approach, estará indo na linha do MDA.

Ah, agora a coisa ficou mais clara. :) Mas veja que conforme você mesmo disse, a UML permite este formalismo (a aproximação a uma linguagem de programação). O que eu queria ver era até que ponto este formalismo me ajudava a entender a modelagem do meu domínio.

> Respondi?

Respondeu sim! Muito obrigado! :)

--
--fx

---

**Fabrício Cabral** <fabriciofx@gmail.com> Mon, Apr 11, 2011 at 10:00 PM
To: UML-BR@yahoogrupos.com.br

Olá MT,

desculpe a demora em lhe responder!

2011/4/4 Marcio Tierno <mtierno@rocketmail.com>

> 1) Como eu disse, a maneira sistemática é: use unidirecionalidade por defninição e bidirecionalidade se absolutamente necessário.

Certo. :)

> 2) Vc está usando uma abordagem ontológica, i.e., baseada no que o objeto "é". Num projeto de software, o ideal é usar uma abordagem funcional, i.e., baseada no que um objeto "faz". Assim, no seu exemplo, não importa muito o que um cérebro ou um corpo são um em relação ao outro mas, sim, o que fazem. O Cérebro precisará "ver" o corpo se precisar enviar-lhe uma mensagem e vice-versa. Aparentemente, trata-se de bidirecionalidade, certo? Bem, um mesmo conjunto de classes terá comportamento diferente para sistemas diferentes. Então, se você tem um sistema no qual vai-se do cérebro para o corpo frequentemente mas só se vai do corpo para o cérebro muito de vez em quando, então há aí uma oportunidade de unidirecionalidade (i.e., economia de design e de recursos computacionais) a ser explorada.

Isto me lembrou de uma palestra que eu assisti há um bom tempo atrás, do prof. Giancarlo Guizzardi (http://www.inf.ufes.br/~gguizzardi). Embora a palestra dele tinha sido sobre ontologias, muitas coisas se encaixam no mundo da UML. E o que mais me chamou a atenção foi que de acordo com uma metodologia apresentada por ele (UFO - Unified Foundational Ontology) existia uma razão porque numa ontologia (e dá pra extender isso pra UML) uma entidade A não se relacionava com uma B, por exemplo.

Se isso lhe interessar, um bom começo seria ler a tese de doutorado dele (que virou um livro): http://www.inf.ufes.br/~gguizzardi/OFSCM.pdf.

> Outro exemplo: relação pai e filho. Um pai tem filhos e um filho obrigatoriamente tem um pai, certo? Então, é bidirecional.
>
> Bom, num sistema de seguro de vida, no qual o filho é o beneficiário, é unidirecional de pai para filho, pois nunca se precisa fazer a navegação em contrário.
>
> Já num sistema de seguro saúde, aí vc pode querer saber quem é o pai a partir do filho, caso este último esteja em atendimento de emergência num hospital e se necessite contatar o titular do plano. Já o contrário, de pai para filho, por incrível que pareceça, será bem pouco necessário nesse sistema e a navegabilidade poderá ser "ao contrário", i.e., de filho para pai mas não de pai para filho. "Lôco", né?

É bem louco sim. :) Mas o exemplo é muito bom, inclusive pra ilustrar

que um mesmo relacionamento entre A e B podem ser bem diferentes de acordo com o domínio onde estas entidades se encontrem. :)

> 5) É isso aí. Por exemplo.

Ah, agora entendi. :)

[]'s

--fx

---

**Fabrício Cabral** <fabriciofx@gmail.com> Mon, Apr 11, 2011 at 10:04 PM
To: UML-BR@yahoogrupos.com.br

Olá Marcelo,

2011/4/6 Marcelo Elias Del Valle <mvallebr@gmail.com>

> MT,
>
> Certo, é isso aí mesmo! Embora a implementação seja uma consequência dos requisitos, a decisão de haver bidirecionalidade ou não é uma decisão de implementação.
> Diga-se de passagem, é muito ruim quando recebemos requisitos tentando impor uma forma de implementar, como por vezes acontece. O correto é deixar essa decisão para o pessoal técnico, ou seja, que programa.

Na minha cabeça ingênua eu via o modelo UML como uma referência, que se o programador estivesse querendo fazer alguma coisa fora deste modelo (como implementar uma bidirecionalidade onde deveria haver uma unidirecionalidade) era um sinal que o programador (ou o modelador) havia feito alguma coisa de errado... :)

Mas obrigado ao Marcelo e o MT pelos esclarecimentos! :)

[]'s

--
--fx

---

**Marcelo Elias Del Valle** <mvallebr@gmail.com> Tue, Apr 12, 2011 at 11:37 AM
To: UML-BR@yahoogrupos.com.br
Cc: Fabrício Cabral <fabriciofx@gmail.com>

Fabrício,

Pelo seu email, você parece bastante interessado no approach do MDA.
Eu não estudei a fundo MDA ainda, mas o Walter Mourão, aqui do grupo, fez um tutorial usando a ferramenta AndroMDA e compartilhou conosco há alguns meses. Segue o link:

https://sites.google.com/site/jgnuteca/

Guardei para fazer quando tiver tempo. Acho que pra você pode ser bem proveitoso também.

Abraços,
Marcelo.

Em 11 de abril de 2011 21:41, Fabrício Cabral <fabriciofx@gmail.com> escreveu:

> Olá Marcelo,
>
> desculpe a demora em responder o seu e-mail. Muitas coisas a fazer... :)

2011/4/4 Marcelo Elias Del Valle <mvallebr@gmail.com>

> A pergunta já mostra que você está entendendo o que está acontecendo.
> A questão é: qual a finalidade do modelo? É preciso lembrar que UML é só a
> linguagem... Ela não determina o que você diz. Eu posso dizer algo em
> português ou em inglês, mas *o que* eu estou dizendo pode ser exatamente a
[Quoted text hidden]
[Quoted text hidden]

--
--fx

[As partes desta mensagem que não continham texto foram removidas]



--
Marcelo Elias Del Valle
http://SorocabaMundi.com - Sorocaba Mundi - Tudo relacionado a Sorocaba
http://www.humansoftware.com.br

---

**Fabrício Cabral** <fabriciofx@gmail.com> Tue, Apr 12, 2011 at 11:52 AM
To: Marcelo Elias Del Valle <mvallebr@gmail.com>

Olá Marcelo, tudo bem? :)

2011/4/12 Marcelo Elias Del Valle <mvallebr@gmail.com>
> Pelo seu email, você parece bastante interessado no approach do MDA.

Na verdade, o grande objetivo desta (e de outras futuras threads) é entender melhor não só a UML mas alguns detalhes de modelagem de sistemas. E, pelo menos na minha cabeça, muitas coisas ainda estão começando a se encaixar e clarear, graças ao pessoal aqui da lista. :)

Sobre o MDA eu já tinha ouvido falar dessa abordagem, inclusive conversando com um amigo que há um bom tempo atrás estava desenvolvendo um sistema de grande porte usando o AndroMDA. Inclusive ele falava de um tal de cartridge (cartucho) que até hoje eu não sei direito pra que serve, mas suspeito que seja a base pra gerar o source do sistema... :)

> Eu não estudei a fundo MDA ainda, mas o Walter Mourão, aqui do grupo, fez um tutorial usando a ferramenta AndroMDA e compartilhou conosco há alguns meses. Segue o link:

> https://sites.google.com/site/jgnuteca/
>
> Guardei para fazer quando tiver tempo. Acho que pra você pode ser bem proveitoso também.

Marcelo, este link é um achado! Muito bom mesmo! Com certeza vou olhar! :)

Abraço!

--
--fx

---

**Marcelo Elias Del Valle** <mvallebr@gmail.com> Tue, Apr 12, 2011 at 12:10 PM
To: Fabrício Cabral <fabriciofx@gmail.com>

Oi Fabrício,

Também achei o link muito bom. Agradecimentos ao Mourão.
Sobre o MDA, a idéia é especificar UML no mesmo nível de formalismo de código, de modo a gerar um "modelo executável". Para executar determinadas tarefas mais dependentes de tecnologia, existe uma forma de conectar o que você cria no modelo com código em uma linguagem de mais baixo nível, como java.

Abraços,
Marcelo.
[Quoted text hidden]
--
Marcelo Elias Del Valle
http://SorocabaMundi.com - Sorocaba Mundi - Tudo relacionado a Sorocaba
http://www.humansoftware.com.br

---

**Marcelo Elias Del Valle** <mvallebr@gmail.com> Wed, Apr 13, 2011 at 2:35 PM
To: UML-BR@yahoogrupos.com.br
Cc: Fabrício Cabral <fabriciofx@gmail.com>

Fabrício,

"*Na minha cabeça ingênua eu via o modelo UML como uma referência, que se o programador estivesse querendo fazer alguma coisa fora deste modelo (como implementar uma bidirecionalidade onde deveria haver uma unidirecionalidade) era um sinal que o programador (ou o modelador) havia feito alguma coisa de errado... :)*"
É possível usar UML desse jeito. Mas nesse caso, o modelo tem de estar de acordo com o código.
Não é assim que a maioria das empresas trabalha, pois na prática isso acaba gerando desperdício de esforço e custo... A maioria prefere deixar esse tipo de decisão para o próprio programador, pois assim o designer não precisa manjar tanto de código.
Um modelo comum em empresas de desenvolvimento é o seguinte: os analistas de requisito e designers criam documentos e diagramas para passar para uma fábrica de software o que o software deve fazer e parte da solução. Nessa etapa usam arquitetos para monitorar se não tem nenhum furo muito grande no funcional. Já vi especificarem, por exemplo, que um sistema web deveria acessar os arquivos da máquina do cliente.
A outra parte da solução deixam por conta da fábrica, que normalmente tem desenvolvedores e arquitetos responsáveis pelo código, onde decidem a melhor maneira de implementar visando reuso, boas práticas, etc.

Minha opinião: existem coisas piores que esse modelo citado acima, mas está longe de ser o ideal. Não gosto do modelo fabril, mas 90% das empresas onde eu trabalho utilizam esse modelo, então não adianta ficar reclamando, temos que nos adaptar e, quando há interesse, sugerir alternativas melhores, como processos ágeis.
Não tenha a ilusão, contudo, de que UML será uma bala de prata, ao menos num curto espaço de tempo. Eu já tive essa ilusão há uns 10 anos e me decepcionei muito. É como SOA: não basta mudar a parte técnica, mudar a parte humana, o "modus operandi", é imprescindível para conseguir utilizar conceitos mais avançados no mundo real. E essas mudanças são lentas. Leeennttaaaaasss.....

Abraços,
Marcelo.

Em 11 de abril de 2011 22:04, Fabrício Cabral <fabriciofx@gmail.com> escreveu:

> Olá Marcelo,
>
> 2011/4/6 Marcelo Elias Del Valle <mvallebr@gmail.com>
>
> > MT,
> >
> > Certo, é isso aí mesmo! Embora a implementação seja uma consequência dos
> > requisitos, a decisão de haver bidirecionalidade ou não é uma decisão de
> > implementação.
> > Diga-se de passagem, é muito ruim quando recebemos requisitos tentando
> > impor uma forma de implementar, como por vezes acontece. O correto é deixar
> > essa decisão para o pessoal técnico, ou seja, que programa.
> >
>
> Na minha cabeça ingênua eu via o modelo UML como uma referência, que se o
> programador estivesse querendo fazer alguma coisa fora deste modelo (como
> implementar uma bidirecionalidade onde deveria haver uma unidirecionalidade)
> era um sinal que o programador (ou o modelador) havia feito alguma coisa de
> errado... :)
>
> Mas obrigado ao Marcelo e o MT pelos esclarecimentos! :)
>
> []'s
>
> --
> --fx
>
> [As partes desta mensagem que não continham texto foram removidas]
>
> 

--
Marcelo Elias Del Valle
http://SorocabaMundi.com - Sorocaba Mundi - Tudo relacionado a Sorocaba
http://www.humansoftware.com.br