From Business Architecture to Software Architecture

Luiz Luz
luizluz@gmail.com

O que será estudado neste captítulo?

- Neste capítulo é apresentado como a modelagem de processo de negócio é utilizada no processo de desenvolvimento de software para definir a arquitetura de software.

Visão Geral

Alguns problemas

- O Modelo de Negócio pode ser utilizado base para modelar e projetar softwares que apoiarão o negócio.
- Porque nem sempre isto é possível ou fácil de fazer?
  - Geralmente o modelo de negócio e o modelo de software são feitos em linguagens diferentes, com técnicas diferentes e utilizando diferentes conceitos.
  - O modelo de negócio e o modelo de software são desenvolvido como dois diferentes projetos e com duas equipes diferentes.
- Por causa desses problemas, a integração entre eles é muito difícil.
- É vantajoso utilizar a mesma linguagem e os mesmos conceitos para modelar o negócio e o software

Visão Geral

Relação entre Modelo de Negócio e Modelo de Software

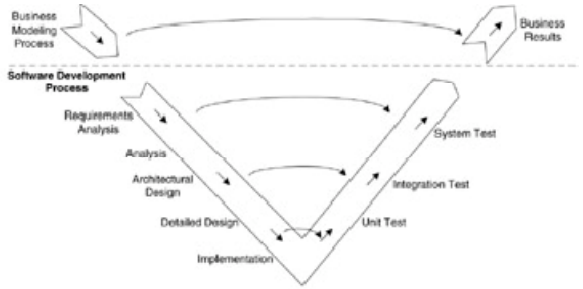
- O modelo de negócio e o modelo de software não possuem um relacionamento “um-para-um”.
  - Muitos elementos do modelo negócio não fazem parte do modelo de software visto que nem todos os processos de negócio serão desenvolvidos ou representados em um software.
  - Vários dos processos contém atividades que são realizadas manualmente, sem precisar da utilização de um software. Estas atividades não farão parte do software.
  - Os conceitos do negócio, como o modelo de metas normalmente também estão fora dos softwares. Da mesma forma, muitos elementos no modelo de software dizem respeito a detalhes técnicos da solução e da construção que não fazem parte do modelo de negócio.

Processo de Desenvolvimento de Software

- Há vários processos de desenvolvimento de software e cada um deles recomenda e utiliza atividades diferentes.
- Há 9 atividades que geralmente fazem parte deles:
  - Análise de Requisitos
  - Análise
  - Desenho Arquitetônico
  - Desenho detalhado
  - Implementação
  - Teste de Unidade
  - Teste de Integração
  - Teste de Sistema
  - Implantação

Processo de Desenvolvimento de Software

- Um processo de software tradicional contém a seguinte anatomia:



Processo de Desenvolvimento de Software

- Os atuais processos de desenvolvimento de software utilizam processos iterativos e incrementais.
  - A cada iteração novas funcionalidades são adicionadas ao software.
  - Nestes casos, para que o primeiro passo aconteça algumas questões já precisam estar definidas, como os requisitos básicos do software.
  - A atividade de requisitos em cada iteração pode detalhar somente os requisitos básicos necessários para esta iteração e, se necessário, levar em consideração as experiências de uma iteração anterior ou, em alguns casos, sugestões de usuários que precisam ser testadas mais cedo.
  - Passando pelas atividades somente uma vez a detecção de erros e de mal entendidos é bem reduzida. Passando por todas as atividades em cada iteração os problemas podem ser encontrados continuamente. Desta forma o projeto pode ser melhor controlado e os erros podem ser corrigidos mais cedo.

Processo de Desenvolvimento de Software

- A modelagem de negócio pode ser feita separadamente da iteração de desenvolvimento, a fim de que a um modelo de negócio estável exista antes de a interação começar.

O que é arquitetura de software?

- O termo arquitetura de software é muito utilizado porém ele não tem uma única definição.
  - "É a estrutura ou estruturas do sistema de computação incluindo componentes de software, as propriedades desses componentes que podem ser visíveis exteriormente e os relacionamentos entre eles."
    - Bass, Len, Paul Clements, and Rick Kazman. *Software Architecture in Practice*. Reading, MA: Addison-Wesley, 1998.]
- A arquitetura de software está relacionada não somente com a estrutura e o comportamento dos componentes, mas também o uso, a funcionalidade, a performance, o desempenho, a reutilização, a inteligibilidade, a restrições econômicas e tecnológicas e negociações, e a questões estéticas.

O que é arquitetura de software?

- Uma arquitetura possui características funcionais, não funcionais e de desenvolvimento, que são utilizadas para avaliá-la.
- Geralmente, realizar a funcionalidade correta na hora certa é considerado o objetivo mais importante de uma arquitetura de software, porém, uma boa arquitetura não pode negligenciar outras questões também importantes.

O que é arquitetura de software?

- As diferentes características de uma arquitetura de software são :
  - **Funcionais**: diz respeito às capacidade de o sistema realizar as funções requisitadas pelos usuários.
  - **Não-funcionais**: diz respeito aos objetivos não funcionais como performance, segurança e outras questões relacionadas à qualidade de uso.
  - **Desenvolvimento**: diz respeito à qualidade do sistema produzido, em relação a modificabilidade, reusabilidade, custo e integridade.

O que é arquitetura de software?

- A arquitetura pode abrange vários sistemas e até mesmo todos os sistemas da empresa.
  - Pode não afetar somente o contexto funcional ou técnico, como também a organização da equipe de desenvolvimento bem como a gestão do projeto.
  - A arquitetura é também base para questões de longo prazo como reuso, linha de desenvolvimento de produtos e modificações futuras no sistema.
  - Quando os sistemas que apóiam o negócio possuem uma arquitetura similar e flexível, é possível realizar, com mais facilidade, mudanças no negócio e no modelo de negócio que esses sistemas apóiam, sem precisar realizar alterações ou grandes alterações nestes sistemas.

O que é arquitetura de software?

- Nem sempre é possível maximizar todas as características da arquitetura.
  - Deve ser estabelecido um balanço para que a arquitetura tenha sucesso.
  - Há interações e dependências entre todas as características da arquitetura que podem gerar conflitos entre elas (Por exemplo: performance x segurança).
- O arquiteto deve pesar a característica e fazer acordos para alcançar o melhor resultado possível.

O que é arquitetura de software?

- Todos os sistemas possuem arquitetura, porém nem sempre ela tem sido construída de maneira planejada
  - Geralmente quando os processos de desenvolvimento de software são dirigidos para escrever códigos a arquitetura de software não é planejada.
- Realmente bons programadores, algumas vezes, podem construir uma visão da arquitetura em suas cabeças, quando a equipe de desenvolvimento é pequena. Porém, quando a equipe de desenvolvimento é grande ou possui programadores com habilidades em diferentes níveis, isto não é possível e o sistema se degenera.

O que é arquitetura de software?

- Uma arquitetura bem definida é o mais importante instrumento para gerir a complexidade de um sistema grande e difícil, e é também a base para a criação de um sistema que pode ser estendido e alterado conforme as necessidades que surgem .
- Uma boa arquitetura ajuda a estruturar tanto o sistema em construção quanto o próprio trabalho realizado na construção do sistema.

Modelando a arquitetura de software

- A utilização da UML para modelar a arquitetura de software é uma técnica bem estabelecida e comumente utilizados.
- Muitos diferentes processos de desenvolvimento concordam com a modelagem usando a UML como linguagem para descrever o software e a sua arquitetura.
- A modelagem da arquitetura de software se dá através de diagramas que são agrupados em visões.

Visões da arquitetura de software

- De maneira similar à arquitetura de negócio, uma arquitetura de software é descrita em com um certo número de visões e cada uma representa um aspecto específico do sistema.
- Há cinco visões de arquitetura de software que têm sido implementadas e utilizadas em muitos projetos de desenvolvimento de software;

Visões da arquitetura de software

- **Caso de Uso**: Ilustra as principais funcionalidades do sistema na percpectiva de um ator externo (pessoas ou outros sistemas) através da comunicação entre os atores e o sistema (diagrama de caso de uso). Geralmente esta visão dirige o desenvolvimento das outras visões.
- **Lógica:** Mostra a realização dos requisitos funcionais através do agrupamento de classes em pacotes (diagrama de classes em pacotes), das classes significativas da arquitetura e de seus relacionamentos (diagramas de classe), do estado e do comportamento das entidades (diagramas de estado), e da colaboração (diagramas de seqüência, diagramas de colaboração, diagramas de atividade).
- **Implantação**: Descreve o sistema em termos dos nodos físico (isto é, computadores e dispositivos) e atribui componentes e processos a estes nodos (diagrama de implantação). Esta visão mostra a topologia do sistema.

Visões da arquitetura de software

- **Implementação:** Estrutura o desenvolvimento do sistema em termos de módulos de código, tanto em código fonte quanto em módulos binários, tais como bibliotecas e arquivos executáveis (componentes gráficos).
- **Processo**: Estruturas do sistema em processos e threads (por exemplo, objetos ativos que executam concorrentemente) e os mecanismos de interação e sincronização entre estes objetos ativos (diagramas de classe, diagramas de seqüência, diagramas de colaboração, diagramas de atividade). A visão de processo também descreve questões específicas, como a inicialização e o finalização do sistema, juntamente com os especiais requisitos não-funcionais, como a falta de tolerância e distribuição de fluxo.

Utilizando a arquitetura de negócio para definir a arquitetura de software

- A transferência das informações no modelo de negócio para o modelo de software não é simples e nem é um processo automático.
  - Infelizmente, não há uma mapeamento um-para-um para esse processo, e não há um simples algoritmo para traduzir o modelo de negócio em um modelo de software.
  - Eles são dois modelos diferentes, com diferentes propósitos.
- O modelo de negócio descreve um negócio ou uma parte específica dele, e nem todas as partes de um negócio são transferida para um sistema de software (por exemplo, as pessoas, as atividades manuais, e muitas metas de negócio).
- Não é possível converter as classes e os objetos do modelo de negócio diretamente para as correspondentes classes e objetos do modelo de software;
  - Isto levaria muitas vezes a uma arquitetura de software muito confusa onde as coisas que não podem ou não devem ir para o sistema de software passa a ser classes e objetos no sistema.

Utilizando a arquitetura de negócio para definir a arquitetura de software

- Quando um modelo de negócios é criado com o objetivo de encontrar requisitos e de ser utilizado como base para um modelo de software, é importante que ele descreve o que é relevante para o modelo de software.
  - Há várias interações e outras questões importantes precisam ser descritas para abordar diretamente a os requisitos de software.
  - Da mesma forma, se o objetivo do modelo de negócio é mais amplo (por exemplo, documentar o negócio ou a identificação de inovações no mundo dos negócios), mais trabalho tem de ser colocado nas partes do modelo de negócios relacionado a estas questões.

Identificando o sistema de informação

- Geralmente o modelo de negócio é utilizado para identificar o sistema de informação mais adequado para o negócio
- Com ele é possível analisar criticamente as necessidades dos negócios e a atual topologia do sistema, de forma a identificar formas de melhorar o suporte para o negócio.
- Nem todos os sistemas no negócio terá de ser recém-desenvolvidos. Normalmente tenta-se minimizar a quantidade de novos desenvolvimento devido ao elevado custo envolvido.

Identificando o sistema de informação

- Existem três tipos de sistemas em uma empresa:
- **Novo**. Sistemas especificamente desenvolvidos para apoiar esta atividade. Eles são desenvolvidos a partir do zero e são otimizados para apoiar os processos.
- **Padrão**. Sistemas de prateleira. As ofertas típicas nesta categoria são os sistemas de contabilidade, fluxo de trabalho, sistemas de administração e pessoal. Embora norma sistemas são muitas vezes mais baratos do que para comprar a desenvolver, mantenha em mente os serviços nestes sistemas são genéricos e não sempre otimizado para o negócio específicas.
- **Antigos Sistemas utilizados no negócio**. Eles representam um investimento anterior e devem ser incorporadas na nova solução, se possível. Em muitos casos, estes sistemas podem ser adaptadas ou estendidos para incluir novos serviços.

Identificando o sistema de informação

- Geralmente utiliza-se o diagrama de processo de negócio e o diagrama de linha de montagem para desenvolver a topologia do sistema.
- Estes diagramas ajudam o arquiteto a identificar atividades que exigem serviços de um sistema de informação. Eles são importantes para identificar os sistemas de informações que são exigidas ou que são utilizados nos processos.

Identificando o sistema de informação

- Normalmente as atividades nos processos de negócio que indicam a utilização dos serviços de sistemas de informação são:
  - Armazenamento, recuperação, organização e administração de informações
  - Processamento, conversão e apresentação de informações
  - Gestão de conhecimento e a tomada de decisões.
  - Comunicação
  - Controle de hardware

Identificando o sistema de informação

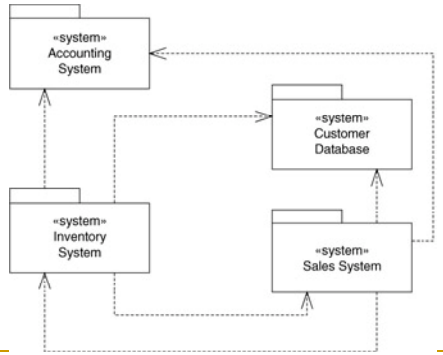
- Não é fácil identificar exatamente quais são os sistemas necessários e quais funcionalidades devem ir para cada sistema, em um processo simples.
- O arquiteto deve equilibrar a funcionalidade de sistemas já utilizados, a funcionalidade do sistema que pode ser comprado dentre os sistemas padrões, e que a funcionalidade é exclusiva ou específica o suficiente para o negócio em questão que torna o desenvolvimento de um novo sistema útil ou necessário.
- Limitações técnicas, tais como os ambientes em que operam os atuais sistemas do negócio pode também ser um fator de decisão entre o que pode ser comprado ou desenvolvido.
- O diagrama de linha de montagem descreve uma lista dos serviços necessários pelos processos, que deverão ser comparados com a atual configuração do sistema e ponderado em relação às restrições de custo.

Identificando o sistema de informação

- Para identificar a estratégia adequada de seleção de sistemas de software um diagrama de topologia de sistema é utilizado.
- Um diagrama de topologia de software é um diagrama de classe com pacotes e as dependências entre pacotes. Cada pacote, estereotipadas de "sistema" (utilizando o padrão UML), denota um sistema neste negócio.
  - Os sistemas podem ser ou já existentes, sistemas padrão ou projetados.
  - As características de cada sistema são documentadas como propriedades dos pacotes.
- De forma iterativa e através de tentativas com diferentes possibilidades de diagramas de topologia de sistema, a topologia de sistema final é descoberta lentamente.
- A outras questões que se pode considerar que estão foram do escopo do livro como a estimar o custo para cada novo sistema ou definição da ordem em que os sistemas existentes serão substituídos.

Identificando o sistema de informação

- Diagrama de Topologia de Sistemas



Procurando Requisitos funcionais

- O mais importante uso do modelo de negócio no contexto da engenharia de software é como base para a identificação dos requisitos funcionais de um sistema, ou seja, casos de uso que o sistema deve fornecer.

Procurando Requisitos funcionais

- Requisitos funcionais do sistema estão descritos em casos de uso que são descrições textuais das da seqüência de interações entre o sistema de informação e o ator externo (o ator pode ser tanto o papel de uma pessoa ou de outro sistema).
- O Caso de uso discute cada etapa da comunicação entre o sistema e seus arredores.
  - Na verdade, este texto especifica a descrição das exigências funcionais, e não o Diagrama de Caso de Uso da UML, que é apenas uma visão geral dos atores e casos de uso presentes no sistema.
- Uma descrição de caso de uso concentra-se em requisitos funcionais, mas às vezes também contém requisitos não funcionais específicos para o caso de uso em questão (por exemplo, questões como performance).
  - Atores no modelo de caso de uso são tipicamente recursos no processo, tais como as pessoas ou máquinas.

Procurando Requisitos funcionais

- Um caso de uso não é equivalente a um processo.
  - Um caso de uso é um serviço que é exigido e que faz parte de um processo fora do sistema.
  - Um caso de uso é plenamente implementado em software, enquanto que um processo, normalmente, é apenas parcialmente implementados em software.
- Um caso de uso pode ser considerado como a especificação dos serviços que o sistema de software fornece ao processo de negócio

Procurando Requisitos funcionais

- Os casos de uso são identificados através de uma análise de caso de uso como parte da atividade de análise de requisitos do processo de desenvolvimento de software.
- A análise de caso de uso utiliza parte do modelo de negócio para identificar os serviços necessários que o sistema de informação deve prover (e que o negócio necessita).
- Recursos (humanos, máquinas, ou outros sistemas) no negócio tornam-se atores para o sistema analisado, e a interação desses recursos na atividade são capturadas em termo de caso de uso.
  - Um recurso é considerado um ator somente em termos de um sistema específico e do ponto de vista da arquitetura de negócio, portanto, não há mudança para o modelo de negócio.

Procurando Requisitos funcionais

- O diagrama de linha de montagem é uma poderosa ferramenta para a identificação dos casos de usos necessários para o sistema.
- Ele descreve as necessárias referências de atividades no âmbito de um processo para pacotes de objetos em um sistema de informação.
  - Um pacote de objetos (representado por um pacote estereotipado para «linha de montagem") pode representar todo um sistema de informação, um subsistema ou componente de um sistema de informação, ou até mesmo uma determinada classe de objetos.

Procurando Requisitos funcionais

- A primeira iteração de análise geralmente começa com referências aos sistemas (cada linha é um sistema de montagem), e é então repetido com um diagrama de linha de montagem específico para cada sistema, onde os pacotes representam subsistemas lógicos desse sistema.
  - Naturalmente, se o objetivo do modelo de negócio é encontrar os requisitos para um sistema específico, você começar diretamente com o diagrama de linha de montagem desse sistema.
  - Os subsistemas e os pacotes são identificados olhando para o modelo de recursos e de informação no modelo de negócio para encontrar divisões lógicas e adequadas das funcionalidade que devem fazer parte do sistema em questão.

Procurando Requisitos funcionais

- As referências dos processos ou atividades para os pacotes de linha de montagem são comunicações requeridas entre recursos(atores) no processo para o sistema de informação.
  - Uma sequência de tais referências torna-se um caso de uso no sistema.

Procurando Requisitos funcionais

- O ponto integração entre os processos de negócio e o caso de uso é o diagrama de linha de montagem.
  - O caso de uso não deve ser definido por uma coleção de referencias de um processo para uma linha de montagem.
  - Um caso de uso deve ser uma funcionalidade completa, que é iniciada por um ator e é retornada de um resultado a partir do sistema.
- Quando utilizar o modelo de negócio para definir casos de uso, você deve olhar para ambas as visões de negócio (perguntando o que é exigido do sistema de informação?) e para a visão de sistema de software (perguntando o que precisa ser feito para tornar a definição do caso de uso completa e bem definida?).
  - Caso contrário, você pode terminar com casos de uso bem pobres, inacabados ou parcial.

Procurando Requisitos funcionais

- Uma parte da concepção do software que é indiretamente afetada pelos requisitos funcionais é a interface gráfica do usuário, o GUI.
  - A GUI pode ser concebida em um estágio inicial de desenvolvimento, após a definição dos atores e dos casos de uso, e, muitas vezes, leva ao desenvolvimento de um protótipo.
  - O protótipo pode então ser colocada nas mãos dos usuários reais, para os seus comentários.
  - A interface de usuário usa a definição do ator como base para a definição dos requisitos funcionais e não funcionais.
  - A interface de usuário é executado pelos objetos de fronteira que lidam com a apresentação, navegação e comunicação de informações entre os atores (a fronteira é objeto, por exemplo, qual um objeto na janela representa a interface do usuário).
- Estes objetos fronteira têm relações com outros objetos que representam as informações comunicadas. A colaboração detalhada entre os objetos de fronteira e os outros objetos são concebidos em detalhe mais tarde e documentado nos diagramas de interação, como os diagramas de seqüência e diagramas de colaboração da UML.

Encontrando requisitos não funcionais

- Requisitos não-funcionais, tais como desempenho, disponibilidade e segurança, não são normalmente ligado a um caso de uso ou a uma funcionalidade específica, em vez disso, eles normalmente são propriedades genéricas que afetam todos ou quase todos casos de uso.
- Muitas vezes os requisitos não funcionais nem sequer são específicos para um determinado sistema, mas são aplicáveis a nível da empresa, ou seja, em todos os sistemas de uma empresa.
- Essas exigências não são normalmente concebidos ou executados em um determinado componente ou subsistema, mas afeta muitos ou todos os componentes e subsistemas.

Encontrando requisitos não funcionais

- O diagrama de processos e as descrições do modelo de negócio são estudados para identificar requisitos não funcionais.
- Requisitos não funcionais são identificados ao olhar para as seguintes necessidades nos processos de negócio:
  - Tempo de atendimento (o tempo entre uma ordem e um produto entregue)
  - Tempo de resposta.
  - Desepenho do processo de negócio monitorado
  - Qualidade medições
  - Disponibilidade
  - Consumo de recursos
  - Segurança;

Encontrando requisitos não funcionais

- Os valores para estes requisitos não funcionais geralmente não estão presentes no diagrama de processos, eles são definidos na atividade de requisitos do processo de desenvolvimento de software.
- Algumas especificações geralmente afetam os requisitos não funcionais de software, que são os modelos de metas.
- Uma boa idéia para ilustrar os requisitos funcionais (como o uso casos) e os requisitos nonfunctional juntos é fazer a matriz abaixo:

	Performance	Availibity	Security	Usability
use case 1	10 ms	98 %	high	n/a
use case 2	2 s	99 %	none	n/a
use case 3	n/a	90 %	n/a	high

Encontrando requisitos não funcionais

- A matriz visualiza claramente que a requisitos não funcionais normalmente não estão ligadas a uma funcionalidade específica, mas abrangem várias funções.
- É importante tanto para descrever os requisitos funcionais e nonfunctional ao sistema e ao subsistema níveis.
- Os requisitos são definidos em um contrato entre o cliente ou usuário do desenvolvimento e organização que é responsável por desenvolver o sistema.