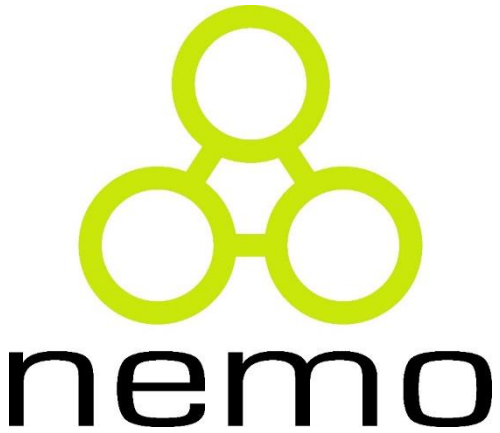
# Engenharia de Requisitos de Software




Jordana S. Salamon

jssalamon@inf.ufes.br

jordanasalamon@gmail.com

DEPARTAMENTO DE INFORMÁTICA

CENTRO TECNOLÓGICO

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

# Qualidade e Agilidade em Requisitos

Revisão?

- Durante o processo de Engenharia de Requisitos, diversos documentos e modelos podem ser elaborados.

- Durante o levantamento de requisitos, requisitos de cliente são capturados em um Documento de Requisitos.

- Na fase de análise, esses requisitos são especificados usando diagramas diversos organizados em dois modelos principais: o modelo estrutural e o modelo comportamental.

# Qualidade e Agilidade em Requisitos

Revisão?

- O modelo conceitual estrutural de um sistema tem por objetivo descrever as informações que esse sistema deve representar e gerenciar.

- O modelo comportamental, por sua vez, provê uma visão ampla do comportamento do sistema.

- No que tange à modelagem do comportamento de um sistema, em um nível superior, os diagramas de casos de uso proveem uma visão externa da funcionalidade do sistema e dos atores nela envolvidos.

nemo

# Qualidade e Agilidade em Requisitos

Revisão?

- O comportamento dos casos de uso é elaborado por meio de descrições de casos de uso. Essas descrições podem ser refinadas por meio de diagramas de atividade ou de sequência.

- Os diagramas de sequência tipicamente mostram visões parciais. Eles, geralmente, são desenvolvidos apenas para fluxos de eventos principais ou são desenvolvidos um diagrama de sequência para o fluxo principal de eventos e diagramas adicionais para os fluxos de exceção e variantes.

nemo

# Qualidade e Agilidade em Requisitos

Revisão?

- Diagramas de atividade permitem consolidar todo esse comportamento em um único diagrama, documentando ramificações, bifurcações e uniões do fluxo de controle.

- Diagramas de sequência são bons para mostrar colaborações entre objetos em um caso de uso.

- Já os diagramas de atividade são úteis para focalizar as atividades de um processo complexo.

nemo

# Qualidade e Agilidade em Requisitos

Revisão?

- Contudo, esses dois tipos de diagramas não são apropriados para observar o comportamento de um objeto isolado ao longo de vários casos de uso. Para tal, são utilizados diagramas de gráfico de estados.

- No contexto da engenharia de requisitos, uma questão importante precisa ser tratada:

- Como fazer a engenharia dos requisitos de um sistema com qualidade, mas ao mesmo tempo de maneira ágil, sem despender tempo elaborando diagramas que acrescentam pouco valor ao projeto?

nemo

# Qualidade e Agilidade em Requisitos

- Não se deve perder de vista que o propósito da modelagem é ajudar a entender o problema de modo que se possa produzir um sistema que atenda às necessidades do usuário.

- Produzir artefatos desnecessários transforma o trabalho de Engenharia de Software em “burocracia de software”.

- Para tratar essa questão, primeiro deve-se considerar a ótica da qualidade.

nemo

# Qualidade e Agilidade em Requisitos

- Sob esse ponto de vista, é fundamental garantir consistência entre os diversos artefatos produzidos.

- Os diversos documentos e modelos proveem diferentes visões de um mesmo sistema e, portanto, devem estar compatíveis entre si.

- É importante observar que os modelos produzidos envolvem conceitos comuns, dentre eles classes, objetos e operações, e que é essencial manter as informações **rastreáveis**.

# Qualidade e Agilidade em Requisitos

- Ambos os modelos, estrutural e comportamental, são necessários para um completo entendimento de um problema.

- Esses modelos se unem para dar forma às classes com atributos, associações e operações, que serão a base para o projeto e a implementação.

- Assim, para uma completa especificação das classes, o que envolve a especificação de suas operações, são necessários os modelos estrutural e comportamental. Além disso, é necessário compará-los para garantir que não há inconsistências entre eles.

nemo

# Qualidade e Agilidade em Requisitos

- Para apoiar a verificação da consistência, técnicas de leitura de modelos de análise podem ser aplicadas.

- Do ponto de vista de agilidade no processo de Engenharia de Requisitos, merece destaque a modelagem ágil.

- A modelagem ágil provê uma série de princípios e valores que podem ser aplicados para nortear a decisão de quais diagramas produzir, de modo que o esforço despendido na análise de requisitos seja compensador.

# Qualidade e Agilidade em Requisitos

- Por fim, visando tanto à qualidade quanto a agilidade, podem ser aplicadas técnicas de reutilização de requisitos.

- A reutilização tende a proporcionar qualidade, uma vez que os requisitos, modelos ou outros artefatos reutilizados já foram avaliados em outros contextos e, por conseguinte, tendem a prover soluções já avaliadas.

- Do ponto de vista da agilidade, ao se reutilizar algum artefato da Engenharia de Requisitos, espera-se que haja um aumento da produtividade, uma vez que não se está partindo do zero.

nemo

# Técnicas de Leitura de Modelos da Análise de Requisitos

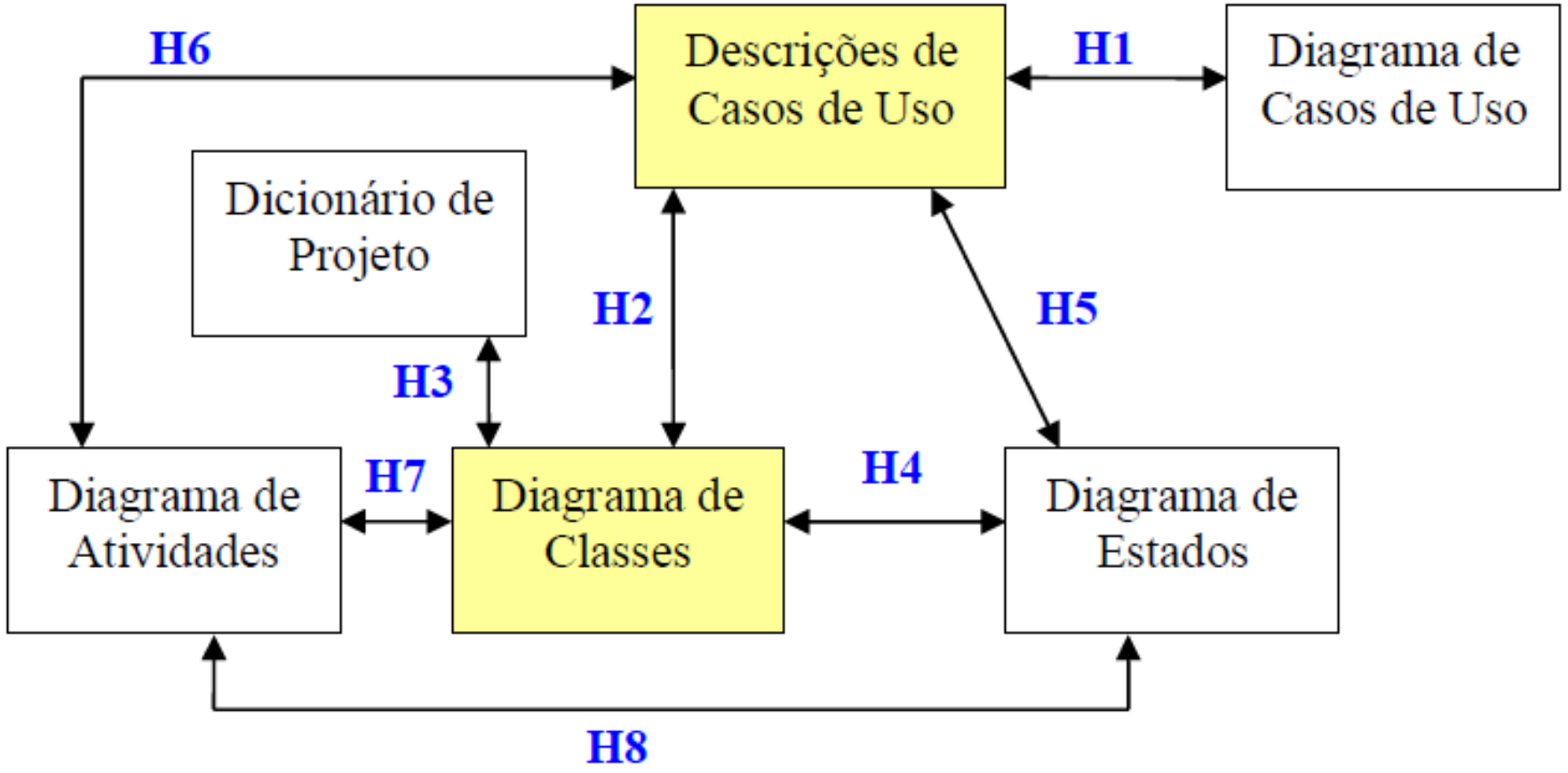
- Na verificação da consistência entre os diversos modelos e diagramas produzidos durante um processo de software orientado a objetos, devem ser consideradas técnicas de leitura de modelos orientados a objetos, as quais podem ser de dois tipos:
  - técnicas para leitura vertical
  - e para leitura horizontal.

- As técnicas para leitura horizontal dizem respeito à consistência entre artefatos elaborados em uma mesma fase, procurando verificar se esses artefatos estão descrevendo consistentemente diferentes aspectos de um mesmo sistema, no nível de abstração relacionado à fase em questão.

nemo

# Técnicas de Leitura de Modelos da Análise de Requisitos

- Técnicas verticais referem-se à consistência entre artefatos elaborados em diferentes fases.



# Técnicas de Leitura de Modelos da Análise de Requisitos

- Essa figura destaca a importância de dois artefatos produzidos na análise de requisitos: as descrições de casos de uso e os diagramas de classes.

- Como se pode notar pela figura, esses dois artefatos têm relações com quase todos os demais, o que mostra o papel central que eles desempenham no processo de desenvolvimento orientado a objetos.

# Técnicas de Leitura de Modelos da Análise de Requisitos

- **H1 – Descrições de Casos de Uso x Diagrama de Casos de Usos**
- a) Para cada caso de uso identificado em um diagrama de casos de uso, deve haver uma descrição de caso de uso associada.
- b) Os nomes dos casos de uso nos dois artefatos devem ser os mesmos.
- c) As descrições dos casos de uso devem fazer menção aos atores envolvidos nos casos de uso e os atores identificados nos dois artefatos devem ser consistentes.
- d) Quando um diagrama de casos de uso apontar uma associação entre casos de uso (inclusão ou extensão), a descrição correspondente deve fazer menção explicitamente à realização do caso de uso associado.

nemo

# Técnicas de Leitura de Modelos da Análise de Requisitos

- **H2 – Descrições de Casos de Uso x Diagrama de Classes**
- a) As classes, associações, atributos e operações modelados no diagrama de classes devem ser necessários e suficientes para realizar cada um dos fluxos de eventos apontados nas descrições de casos de uso.
- b) Quando uma descrição de caso de uso fizer menção a dados de uma classe no diagrama de classes, então a descrição do caso de uso deve ser consistente com os atributos modelados na correspondente classe do diagrama de classes (e vice-versa).
- c) Para manter os modelos rastreáveis, a descrição de caso de uso deve enumerar as classes envolvidas em sua realização.

# Técnicas de Leitura de Modelos da Análise de Requisitos

- **H3 – Dicionário de Projeto x Diagrama de Classes**
- a) Para cada classe existente no diagrama de classes deve haver uma entrada no dicionário de projeto, associada a uma descrição sucinta da mesma.
- b) Todos os atributos e operações apresentados no diagrama de classes devem estar enumerados e sucintamente descritos na entrada do dicionário de projeto referente à correspondente classe.
- c) Restrições de integridade não passíveis de registro pela notação da UML devem ser explicitamente declaradas no modelo de classes. As classes, associações e atributos envolvidos em uma restrição de integridade devem estar consistentes com o diagrama de classes e suas descrições no dicionário de projeto.

nemo

# Técnicas de Leitura de Modelos da Análise de Requisitos

- **H4 – Diagrama de Estados x Diagrama de Classes**

- **a**) Um diagrama de estados deve estar relacionado a uma classe modelada no diagrama de classes.
- b) A classe correspondente no diagrama de classes deve conter atributos, associações e/ou operações capazes de indicar todos os estados pelos quais um objeto da mesma pode passar.

nemo

# Técnicas de Leitura de Modelos da Análise de Requisitos

- **H5 – Diagrama de Estados x Descrições de Casos de Uso**

- a) Os eventos associados a transições entre estados em um diagrama de estados devem corresponder a fluxos de eventos ou a casos de uso em uma descrição de casos de uso.
- b) Quando pertinente, a descrição de caso de uso deve fazer uma menção explícita à transição para o novo estado.

# Técnicas de Leitura de Modelos da Análise de Requisitos

- **H6 – Diagrama de Atividades x Descrições de Casos de Uso**

- **a) Um diagrama de atividades deve mostrar as atividades no contexto de um fluxo de eventos descrito em uma descrição de casos de uso. Cursos alternativos (de exceção ou variantes) devem ser mostrados no mesmo diagrama de atividades.**
- **b) A sequência de atividades em um diagrama de atividades deve estar consistente com a sequência de passos da descrição do caso de uso correspondente.**

nemo

# Técnicas de Leitura de Modelos da Análise de Requisitos

- **H7 – Diagrama de Atividade x Diagrama de Classes**

- • Todos os objetos em um diagrama de atividades devem ter a indicação de a qual classe pertencem. Essas classes devem estar modeladas no diagrama de classes.
- • Algumas atividades em um diagrama de atividades, tipicamente aquelas que apontam um processamento a ser feito pelo sistema, podem indicar a necessidade de uma operação na classe de um dos objetos que são entrada para essa atividade. Quando este for o caso, a operação deve ser mostrada na classe correspondente do diagrama de classes.

nemo

# Técnicas de Leitura de Modelos da Análise de Requisitos

- **H8 – Diagrama de Atividade x Diagrama de Estados**

- a) Quando um objeto em um diagrama de atividades tiver o seu estado indicado e houver um diagrama de estados para a classe desse objeto, então os estados devem estar consistentes, inclusive em relação ao evento que altera o estado do objeto.

# Modelagem Ágil

- Ao contrário da modelagem conceitual estrutural em que o modelo de classes é basicamente o modelo a ser construído, na modelagem comportamental há diversos modelos e diagramas que podem ser empregados (modelos de casos de uso, diagramas de estados, diagramas de sequência e diagramas de atividades).

- Cada um desses diagramas trabalha uma perspectiva diferente e, portanto, diferentes diagramas podem e devem ser elaborados para prover uma visão abrangente do comportamento de um sistema.

- Entretanto, é fundamental considerar a relação custo-benefício do uso desses modelos. Neste contexto, é importante levar em consideração princípios da Modelagem Ágil.

# Modelagem Ágil

- A Modelagem Ágil é uma coleção de valores, princípios e práticas de modelagem de software que pode ser aplicada a projetos de desenvolvimento de software de modo a torná-los mais leves e efetivos.

- Dentre os princípios de modelagem propostos, destacamos os seguintes:

- • **O sistema de software é seu objetivo principal**; possibilitar o próximo trabalho é seu objetivo secundário: o principal objetivo de um projeto de desenvolvimento de software é produzir um sistema de software de qualidade e não criar uma documentação.

nemo

# Modelagem Ágil

- Contudo, durante o desenvolvimento, é necessário preparar o terreno para a próxima atividade, que no caso da modelagem de análise pode ser a fase de projeto ou mesmo de manutenção.

- Para apoiar atividades subsequentes do processo de software, será necessário ter também uma documentação suficiente e de qualidade.

- • **Modele com um propósito**: para que um certo diagrama seja elaborado, deve-se ter uma finalidade em mente.

- Um diagrama deve ser útil para comunicar informações para clientes e usuários, ajudando a se obter um entendimento comum dos requisitos, ou deve ajudar desenvolvedores a compreender melhor algum aspecto de software.

nemo

# Modelagem Ágil

- Com base na meta estabelecida, deve-se escolher o tipo de diagrama mais apropriado para atingir a meta e o nível de detalhe a ser utilizado.

- • **Use múltiplos modelos**: há muitos modelos / diagramas e níveis de descrição (notação) que podem ser usados descrever sistemas de software.

- Contudo, normalmente apenas um pequeno subconjunto deles é essencial para a maioria dos projetos. Assim, para que a elaboração de um certo diagrama se justifique, ele deve adicionar valor ao projeto.

- Ou seja, apenas aqueles diagramas que ofereçam valor à sua audiência alvo devem ser usados.

nemo

# Modelagem Ágil

- • **Diminua a carga de trabalho (Viaje leve)**: cada um dos modelos e diagramas elaborados, se conservado ao longo do desenvolvimento, precisa ser mantido à medida que mudanças nos requisitos acontecem.

- Isso representa trabalho para a equipe e, portanto, toda vez que se decide conservar um modelo, está-se comprometendo a agilidade em prol da conveniência de se ter aquela informação disponível para a equipe.

- A diretriz, portanto, é conservar apenas aqueles modelos que fornecerão valor em longo prazo e descartar o restante. Esse princípio tem várias consequências:

nemo

# Modelagem Ágil

- o Deve-se selecionar apenas um pequeno conjunto de modelos e diagramas para serem elaborados, mantendo-se em linha com o princípio anterior;

- o Caso se opte por elaborar um certo diagrama para compreender melhor um certo aspecto do software, ele pode ser elaborado, mas não precisa ser necessariamente incorporado à documentação.

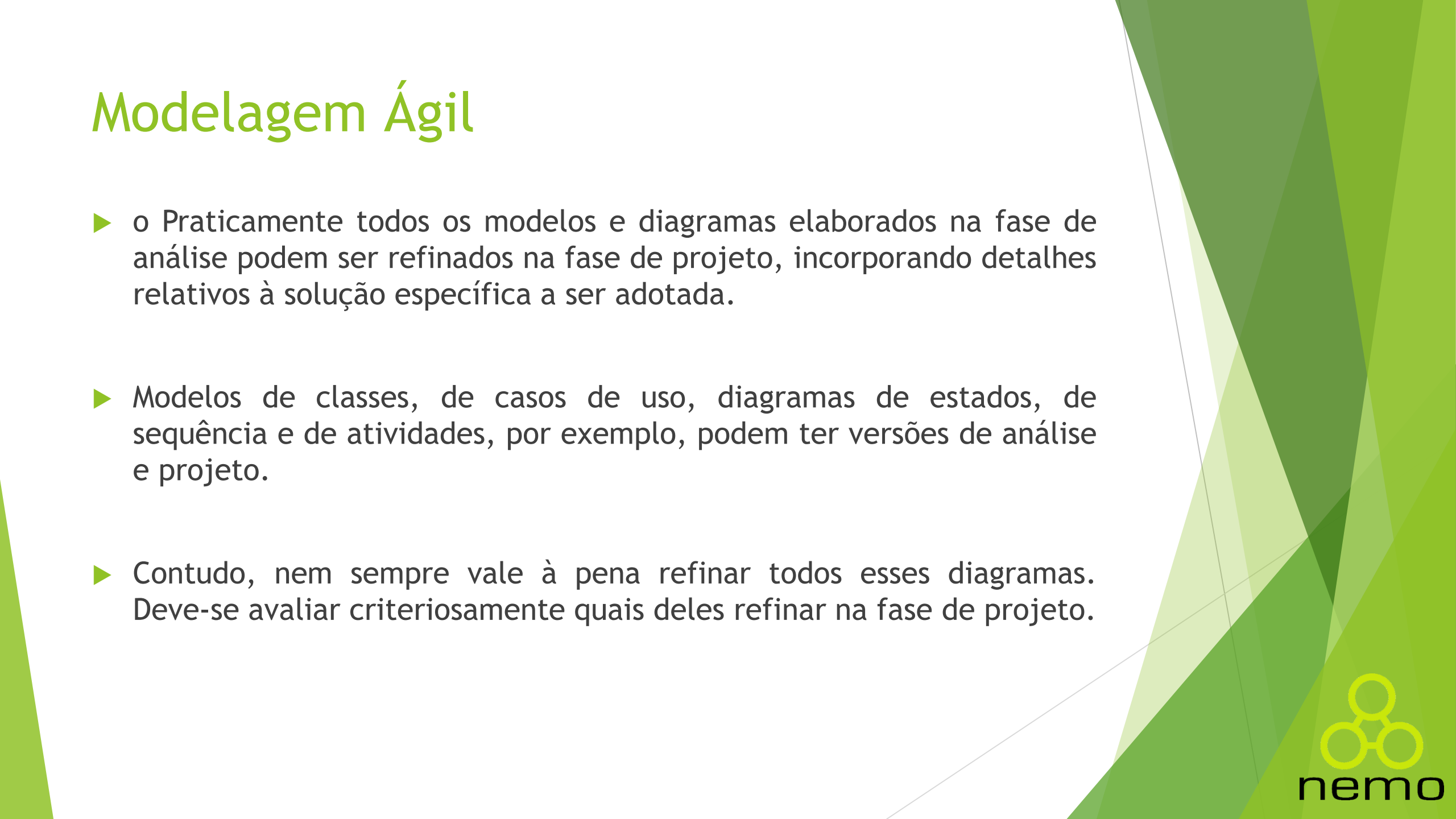
- Às vezes, um esboço em uma folha de papel cumpre o propósito e o diagrama pode ser descartado em seguida;

# Modelagem Ágil

- o Praticamente todos os modelos e diagramas elaborados na fase de análise podem ser refinados na fase de projeto, incorporando detalhes relativos à solução específica a ser adotada.

- Modelos de classes, de casos de uso, diagramas de estados, de sequência e de atividades, por exemplo, podem ter versões de análise e projeto.

- Contudo, nem sempre vale à pena refinar todos esses diagramas. Deve-se avaliar criteriosamente quais deles refinar na fase de projeto.



# Modelagem Ágil

- • **Adote a simplicidade**: pressuponha que a solução mais simples é a melhor e mantenha os modelos tão simples quanto puder.

- Não modele seu sistema em excesso hoje, mostrando características adicionais não requeridas.

- • **Conheça os modelos e as ferramentas usadas para criá-los**: Entenda o propósito de cada modelo / diagrama, seus pontos fortes e fracos e as principais notações a serem empregadas em cada fase do processo de desenvolvimento.

- Conheça também as ferramentas usadas para criá-los e suas limitações. Isso dará agilidade ao desenvolvimento de modelos.

nemo

# Modelagem Ágil

- Ao considerar esses princípios na modelagem segundo o paradigma orientado a objetos, chega-se ao seguinte conjunto de orientações:

- a) Modelos de casos de uso e de classes são essenciais para o desenvolvimento e devem ser elaborados durante a análise de requisitos.

- Descrições de casos de uso completas só devem ser elaboradas para casos de uso mais complexos, tipicamente aqueles relacionados aos principais processos de negócio sendo apoiados pelo sistema em desenvolvimento.

nemo

# Modelagem Ágil

- b) Diagramas de estados só devem ser elaborados para classes com comportamento dependente do estado.

- Recomenda-se elaborar diagramas de estados apenas para as classes que possuírem três ou mais estados relevantes.

- 

- Se julgado útil para a compreensão, pode ser elaborado um diagrama de estados para uma classe com apenas dois estados relevantes.

- Entretanto, deve-se avaliar se o mesmo não pode ser descartado após cumprir seu propósito, não sendo incorporado à documentação do projeto.

nemo

# Modelagem Ágil

- c) Diagramas de atividades só devem ser elaborados quando forem úteis para refinar o entendimento provido pela descrição de um caso de uso complexo, normalmente com várias atividades paralelas, iterativas, condicionais ou alternativas.

- Diagramas de atividades também são uma boa opção para a modelagem de processos de negócio.

- d) Em relação ao refinamento de modelos e diagramas na fase de projeto, de maneira geral, modelos de classe devem ser refinados, uma vez que eles são a base para a implementação.

- O mesmo não ocorre com os modelos comportamentais (modelos de casos de uso, diagramas de estados, sequência e atividades), os quais, na maioria das vezes, basta manter a versão de análise.

# Reutilização na Engenharia de Requisitos

- A reutilização no desenvolvimento de software tem como objetivos melhorar o cumprimento de prazos, diminuir custos e obter produtos de maior qualidade.

- Artefatos de software são reutilizados com o intuito de diminuir o tempo de desenvolvimento, investindo-se esforço na sua adaptação ao invés de se investir esforço na sua construção a partir do zero.

- Ao longo do processo de software, diversos tipos de artefatos podem ser reutilizados, dentre eles modelos, especificações, planos, código-fonte etc.

nemo

# Reutilização na Engenharia de Requisitos

- Analisando o processo de Engenharia de Requisitos, é possível notar que a reutilização pode ser útil, sobretudo no reúso de requisitos de sistemas similares e de modelos conceituais.

- Contudo, na prática, na grande maioria das vezes, requisitos e modelos são construídos a partir do zero.

- Ou seja, requisitos de cliente são levantados junto aos interessados e posteriormente refinados em requisitos de sistema, quando modelos são construídos levando-se em conta os requisitos inicialmente capturados.

- Entretanto, essa abordagem tem se mostrado insuficiente, pois desconsidera a reutilização do conhecimento já existente na organização acerca do domínio do problema.

nemo

# Reutilização na Engenharia de Requisitos

- Ao especificar requisitos para um sistema é interessante ter em mente que muito provavelmente alguém já desenvolveu algum sistema muito parecido ou até mesmo idêntico ao que está sendo desenvolvido.

- Assim, caso se tenha acesso aos requisitos de um sistema similar ao que será desenvolvido, certamente o esforço para se obter requisitos para o mesmo será bem menor.

- O reúso na Engenharia de Requisitos tem por objetivo a utilização de requisitos (e outros artefatos relativos às fases da Engenharia de Requisitos) de projetos anteriores com o intuito de auxiliar a obtenção dos requisitos para um novo sistema a ser desenvolvido.

# Reutilização na Engenharia de Requisitos

- Requisitos podem ser reutilizados de diversas maneiras, dentre elas pelo reúso de especificações de projetos similares ao projeto atual, informalmente através da experiência das pessoas ou pela reutilização de artefatos desenvolvidos previamente com vistas ao reúso.

- Por exemplo, ao se analisar um projeto similar, desenvolvido para um mesmo domínio de aplicação, pode-se concluir que diversos requisitos funcionais, não funcionais e até regras de negócio podem se aplicar ao novo projeto em desenvolvimento.

- Neste caso, os requisitos podem ser copiados, e adaptados, quando necessário, de um projeto para o outro.

nemo

# Reutilização na Engenharia de Requisitos

- Outra abordagem bastante utilizada pelas organizações consiste em definir equipes que atuam sistematicamente em um certo domínio de aplicações, de modo que seus membros possam reutilizar o conhecimento que têm sobre esse domínio no desenvolvimento de novos projetos.

- Ainda que essas abordagens mais informais e oportunistas tragam alguns benefícios, elas não exploram adequadamente as potencialidades de uma reutilização sistemática.

- Em uma abordagem sistemática de desenvolvimento para e com reutilização, primeiramente artefatos são concebidos explicitamente para serem reutilizados.

nemo

# Reutilização na Engenharia de Requisitos

- Depois, os projetos de desenvolvimento reutilizam esses artefatos, fazendo as adaptações necessárias.

- Uma vez que esses itens tenham sido desenvolvidos para reúso, o esforço de adaptação e reutilização tende a ser menor. Assim, para se obter os reais benefícios da reutilização, é importante que os artefatos já sejam desenvolvidos pensando no reúso posterior.

- Neste contexto, merecem destaque três abordagens, discutidas na sequência: Engenharia de Domínio, Ontologias e Padrões de Análise.

nemo

# Engenharia de Domínio

- A Engenharia de Domínio representa um enfoque sistemático para a produção de componentes reutilizáveis que engloba atividades de análise, projeto e implementação de domínio, as quais objetivam, respectivamente:

- representar requisitos comuns de uma família de aplicações por meio de modelos de domínio,

- disponibilizar modelos arquiteturais para aplicações a partir de um único modelo de domínio;

- disponibilizar implementações de componentes que representam funcionalidades básicas de aplicações relacionadas a um domínio.

nemo

# Engenharia de Domínio

- A Análise de Domínio é a atividade diretamente ligada à reutilização na Engenharia de Requisitos.

- A Análise de Domínio visa capturar os elementos relevantes de um domínio de aplicações e disponibilizá-los para serem utilizados no desenvolvimento de diferentes sistemas de apoio a negócios neste domínio.

- Assim, a Análise de Domínio busca explicitar e modelar aspectos de domínio, produzindo artefatos (tipicamente modelos) que contêm informações sobre o domínio e que podem ser reutilizados no desenvolvimento de sistemas.

# Engenharia de Domínio

- Pode-se fazer um paralelo entre a Análise de Domínio e a Análise de Requisitos: ambas enfocam a modelagem conceitual de um domínio de aplicações.

- Entretanto, enquanto a análise de requisitos convencional enfoca a modelagem dos aspectos do domínio que são relevantes para um sistema específico, a análise de domínio é mais abrangente e visa capturar elementos de informação do domínio potencialmente relevantes para o desenvolvimento de diversos sistemas neste domínio, estando, portanto, em nível mais alto de abstração.

- Em outras palavras, ao invés de explorar requisitos de uma aplicação específica, na análise de domínio, os requisitos explorados dizem respeito a uma família de aplicações de uma determinada área.

nemo

# Engenharia de Domínio

- Neste sentido, pode-se considerar que a Análise de Domínio direciona a Engenharia de Requisitos, pois seus modelos de domínio, mais abstratos, fornecem uma base para o trato com requisitos no contexto de um projeto específico.

- Para se fazer uma análise de domínio, pode-se trabalhar de maneira análoga à análise de requisitos convencional, consultando-se especialistas do domínio e modelando o conhecimento capturado, de modo que esse conhecimento possa ser reutilizado nos vários projetos que façam parte desse domínio.

# Engenharia de Domínio

- Há, ainda, diversos métodos de análise de domínio, os quais procuram explorar as especificidades dessa atividade, dentre eles FODA, ODM, EDLC, FORM, RSEB e Catalysis.

- Contudo, qualquer que seja a abordagem de análise de domínio empregada, deve-se ter em mente que um modelo de domínio deve conter os elementos comuns às várias aplicações do domínio e não detalhes específicos de uma ou de todas as aplicações.

- Assim, para ser realmente reutilizável, um modelo de domínio deve ser mais geral e abstrato, contendo o conhecimento de senso comum a respeito do domínio, sem incluir detalhes que podem ser relevantes apenas para uma ou para poucas aplicações.

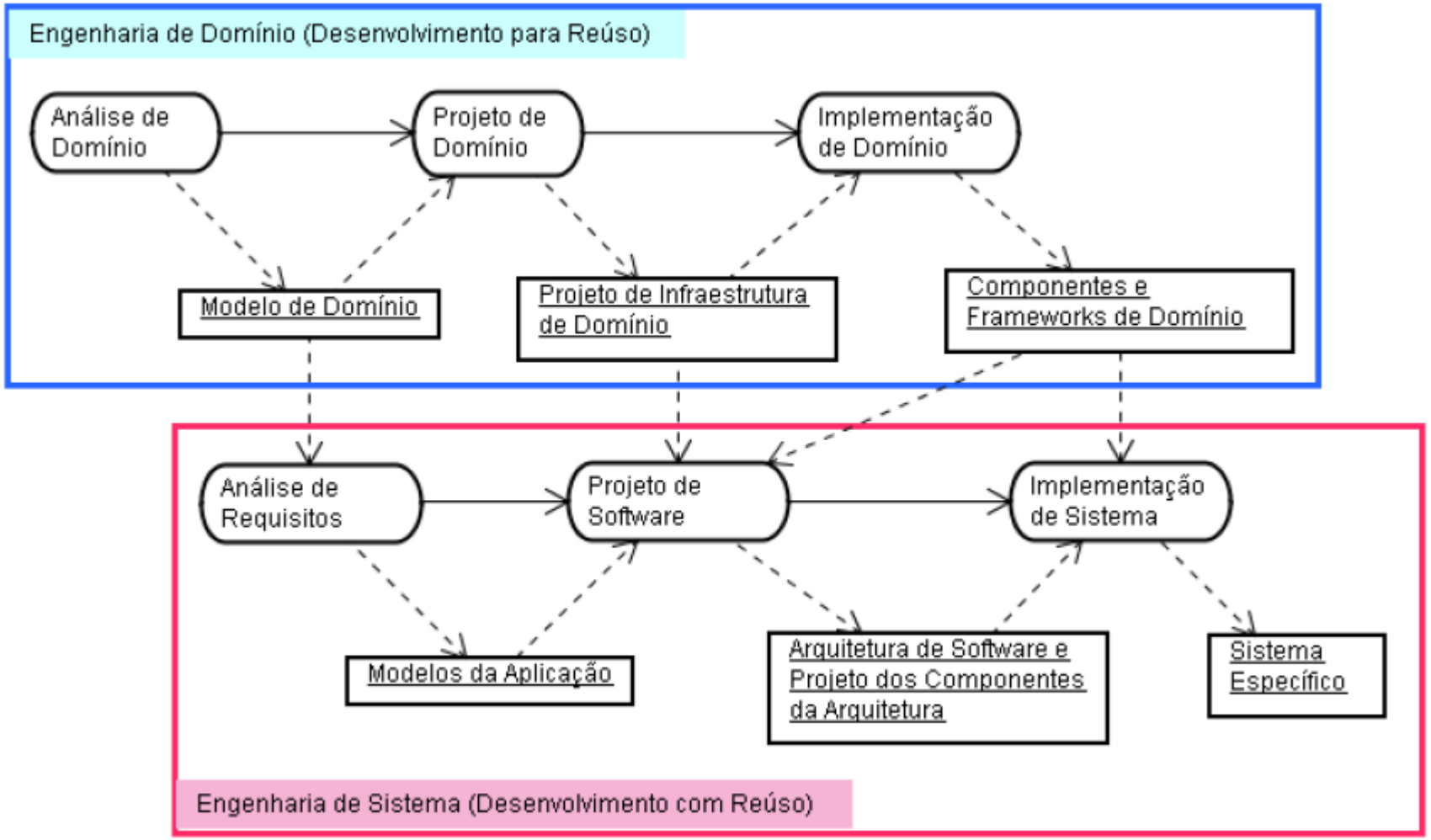
# Engenharia de Domínio

- Deve-se apontar, ainda, que a análise de domínio apenas será uma abordagem interessante, caso haja uma real necessidade de reúso no domínio em questão.

- A análise de domínio é uma atividade complexa que consome tempo e recursos, podendo ser anterior ou paralela a um processo de desenvolvimento de software.

- Assim, se não houver uma real possibilidade de reúso nesse domínio, a análise de domínio torna-se inviável.

# Engenharia de Domínio




nemo

# Ontologias de Domínio

- Uma grande dificuldade atual no desenvolvimento de sistemas para apoiar processos de negócio em um mesmo domínio, sobretudo quando os mesmos precisam trabalhar de maneira integrada, é a existência de diversas visões parciais sobre esse domínio.

- Cada visão parcial carrega consigo um vocabulário e determinados valores próprios, dificultando a integração entre os diversos profissionais pela ausência de padronização.

- Esse problema pode ser minimizado se a conceituação envolvida no domínio for, pelo menos parcialmente, explicitada, o que pode ser feito por meio de uma ontologia de domínio.

# Ontologias de Domínio

- Uma ontologia de domínio é um artefato de engenharia que busca explicitar a conceituação de um domínio particular.

- Uma conceituação, por sua vez, corresponde ao conjunto de conceitos usados para interligar abstrações de entidades de um dado domínio.

- Assim, uma ontologia de domínio tem como objetivo explicitar e formalmente definir os conceitos, relações, propriedades e restrições em um domínio particular.

nemo

# Ontologias de Domínio

- Ontologias de domínio têm diversos usos, dentre eles:
- (i) apoio à comunicação entre pessoas envolvidas no domínio;
- (ii) integração de dados e interoperabilidade de sistemas desenvolvidos para o domínio e
- (iii) especificação reutilizável para a construção de sistemas no domínio.

- Ontologias de domínio podem ser usadas como modelos de domínio em uma abordagem de Engenharia de Domínio.

- Assim, pode-se pensar que o processo de Engenharia de Domínio, neste caso, envolveria atividades de Modelagem da Ontologia, Projeto da Ontologia e Implementação da Ontologia.

nemo

# Ontologias de Domínio

- Do ponto de vista da Engenharia de Requisitos, está-se mais interessado em ontologias como modelos conceituais.

- Assim, o foco está nas ditas ontologias de referência, as quais representam um modelo de consenso dentro de uma comunidade.

- Uma ontologia de referência é uma especificação independente de solução, com o objetivo de fazer uma descrição clara e precisa de entidades do domínio, para propósitos de comunicação, aprendizado e resolução de problemas.

nemo

# Ontologias de Domínio

- Vale a pena destacar similaridades e diferenças entre modelos de domínio em uma abordagem tradicional de Engenharia de Domínio e em uma abordagem baseada em ontologias.

- Ambos têm como propósito capturar a conceituação de um certo domínio, estando em um nível de abstração mais elevado do que um modelo conceitual de um sistema.

- Entretanto, ontologias são desenvolvidas com vários propósitos em mente (e não somente de servir como uma especificação base para o desenvolvimento de sistemas), o que faz com que tenha de ser especificada com mais rigor, normalmente exigindo o consenso em uma comunidade, tornando-a mais geral do que modelos de domínio.

nemo

# Ontologias de Domínio

- Uma vez que uma ontologia é um artefato complexo de engenharia, ela deve ser construída usando métodos apropriados.

- Há diversos métodos existentes para o desenvolvimento de ontologias, dentre eles SABiO, Neon e Methontology .

nemo

# Padrões de Análise

- Padrões de software formalizam soluções para problemas recorrentes no desenvolvimento de software, com base na experiência de especialistas, permitindo que essas soluções possam ser reutilizadas em várias outras situações, por outros especialistas.

- Na engenharia de software, padrões são usados para descrever soluções de sucesso para problemas de software comuns.

- Esses padrões refletem a estrutura conceitual dessas soluções e podem ser aplicados várias vezes na análise, projeto e produção de aplicações, em um contexto particular.

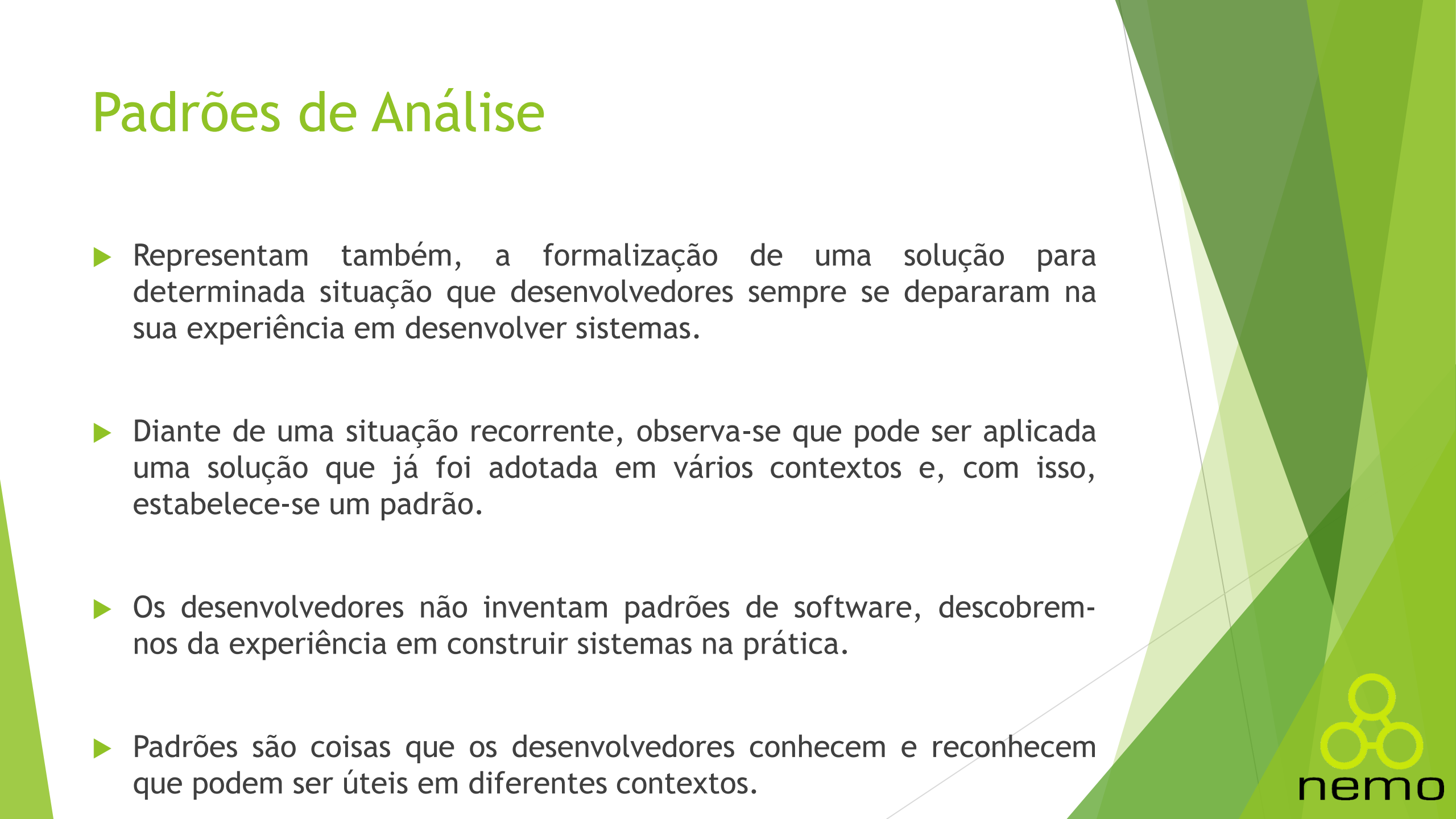
# Padrões de Análise

- O uso pelos engenheiros de software de soluções já conhecidas e testadas ajuda a aumentar o nível de abstração, a produtividade e a qualidade dos projetos nas várias fases do desenvolvimento de sistemas.

- No contexto de desenvolvimento de software, padrões procuram representar a experiência de muitos esforços de reengenharia de desenvolvedores que tentaram adquirir maior reusabilidade e flexibilidade em seus produtos de software.

# Padrões de Análise

- Representam também, a formalização de uma solução para determinada situação que desenvolvedores sempre se depararam na sua experiência em desenvolver sistemas.
- Diante de uma situação recorrente, observa-se que pode ser aplicada uma solução que já foi adotada em vários contextos e, com isso, estabelece-se um padrão.
- Os desenvolvedores não inventam padrões de software, descobrem-nos da experiência em construir sistemas na prática.
- Padrões são coisas que os desenvolvedores conhecem e reconhecem que podem ser úteis em diferentes contextos.



# Padrões de Análise

- Padrões existem em várias fases do desenvolvimento de software.
- Para a Engenharia de Requisitos, os padrões de interesse são os padrões de análise.
- Padrões de análise são definidos a partir de modelos conceituais de aplicações.
- Ao se criar um modelo conceitual, os analistas percebem que muitos aspectos de projetos anteriores se repetem.
- Se ideias usadas em projetos anteriores são úteis atualmente, então, eles melhoram essas ideias e as adaptam para atender a uma nova demanda.

nemo

# Padrões de Análise

- Para reutilizar um padrão de análise em outra aplicação, é preciso reinterpretar cada classe no padrão como uma classe correspondente no novo sistema.
- A estratégia é a abstração do modelo inicial, de onde novos modelos podem ser desenvolvidos.
- Padrões de análise podem ser referidos como grupos de conceitos que representam uma construção comum na modelagem de negócios.
- Eles podem ser relevantes para somente um domínio ou podem ser expandidos para vários domínios.

# Padrões de Análise

- Os padrões de análise lidam com problemas gerais de modelagem, sendo apresentados através de um problema particular em um domínio, onde é mais fácil de entendê-los.
- Conhecendo-os, são identificadas inúmeras situações onde aplicá-los.
- Padrões de maneira geral são registrados em catálogos.
- Um catálogo de padrões é uma coleção de padrões com uma estrutura e uma organização.
- Os padrões são subdivididos em categorias e há relações entre eles.

nemo

# Padrões de Análise

- A classificação em um catálogo facilita o aprendizado e direciona os esforços para encontrar novos padrões, torna-os úteis para os leitores, que podem lembrar dos padrões, identificá-los em diferentes situações e reutilizá-los no desenvolvimento de algum sistema.

- Um dos catálogos de padrões de análise mais conhecidos é o descrito por Fowler em seu livro Analysis Patterns: Reusable Objects Models (Padrões de Análise: Modelos de Objetos Reutilizáveis), onde há mais de 40 padrões de análise, contendo modelos já aplicados em projetos nos quais o autor trabalhou.

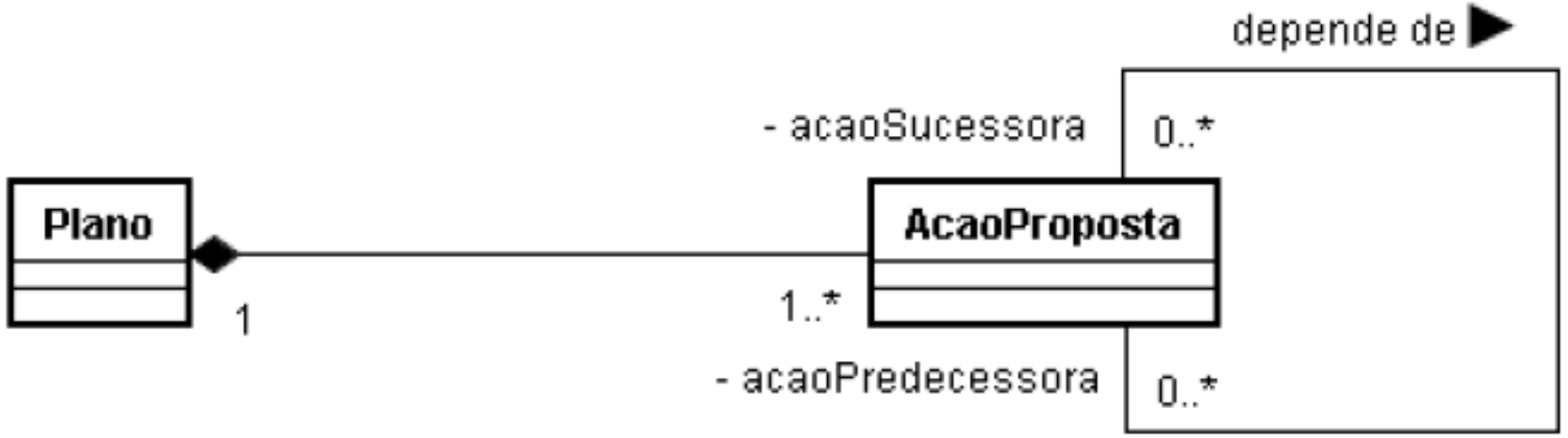
# Padrões de Análise

- Os padrões neste catálogo são organizados em dez categorias principais, a saber: Responsabilidades, Observações e Medições, Observações para Finanças Coorporativas, Referência a Objetos, Inventário e Contabilidade, Uso de Modelos de Contabilidade, Planejamento, Negociação, Contratos Derivados e Pacotes de Negociação.
- Exemplos de modelos estruturais propostos em padrões de análise:


nemo

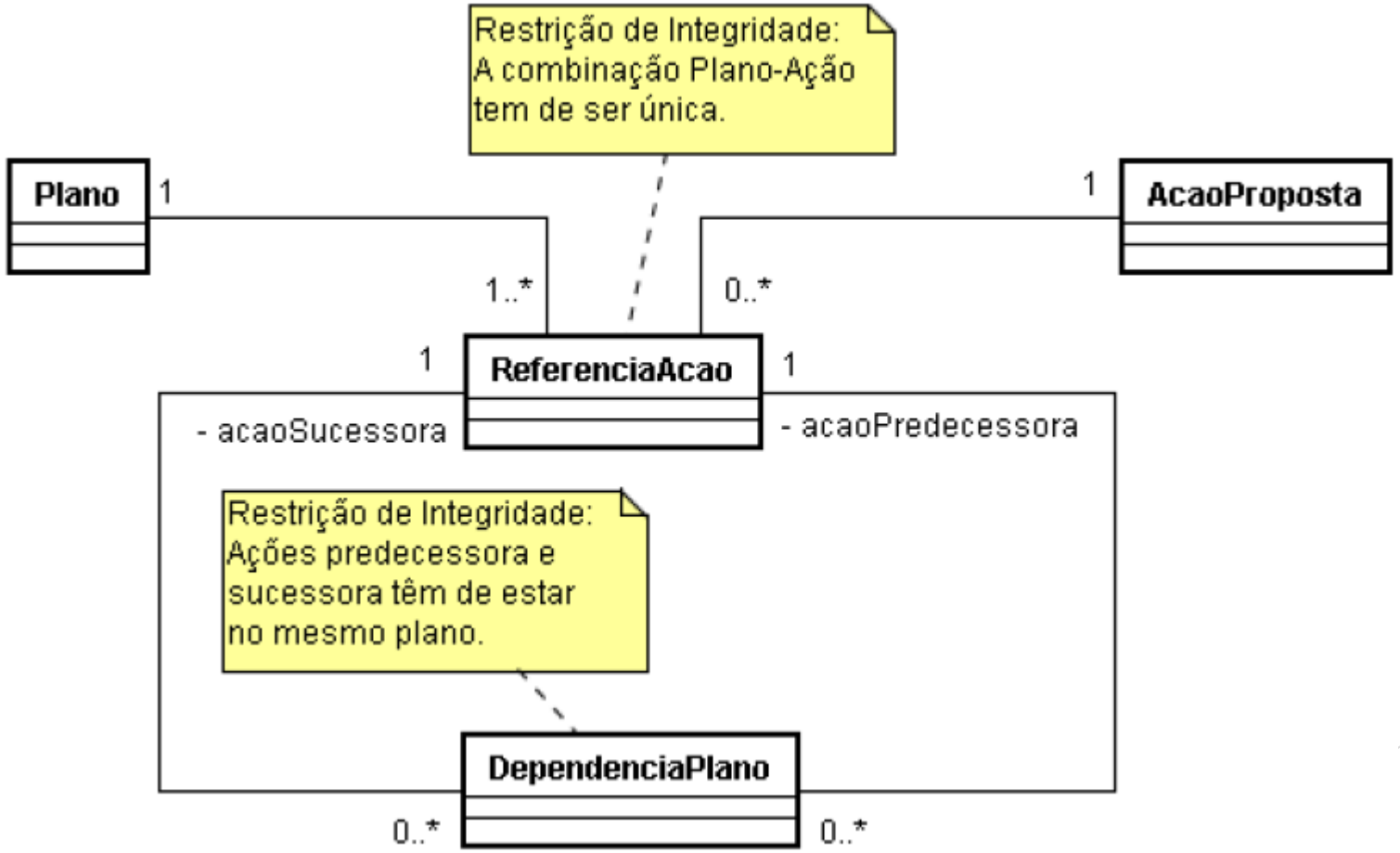
# Padrões de Análise

- Exemplos de modelos estruturais propostos em padrões de análise:



# Padrões de Análise

- Exemplos de modelos estruturais propostos em padrões de análise:



# Padrões de Análise

- Utilizar ontologias e padrões de análise para a modelagem conceitual de sistemas abre espaço para uma visão mais abrangente do domínio tratado, levando a um melhor entendimento do mesmo.

- Isso pode diminuir o tempo gasto na especificação de requisitos, pois os analistas já têm uma fonte preliminar para aprender sobre o domínio, que estabelece uma terminologia comum aos especialistas do negócio.

- Padrões de Análise, assim como ontologias, descrevem aspectos no nível de conhecimento.

- Assim, uma vez que eles proveem conhecimento sobre soluções bem sucedidas a problemas recorrentes no desenvolvimento de software, eles favorecem a reutilização. No entanto, ontologias capturam a estrutura conceitual intrínseca de um domínio, enquanto padrões de análise focalizam a estrutura de uma aplicação.

# That’s all Folks!