Variabilidade em LPS

Documentação e gerenciamento da variabilidade como uma das principais atividades que caracterizam a engenharia da linha de produtos de software (ELPS).

Prof. Sylvio Souza

1. Itens iniciais

Propósito

A variabilidade serve para apoiar o desenvolvimento e o reaproveitamento de artefatos de desenvolvimento variável. Na ELPS, a variabilidade é uma propriedade essencial dos artefatos de domínio. Assim, a modelagem de variabilidade é usada para capturar a variabilidade dos requisitos de domínio, arquitetura, componentes e testes.

Objetivos

- Reconhecer os princípios da variabilidade da linha de produtos de software (LPS).
- Aplicar a variabilidade de uma LPS em artefatos de requisitos e projeto.
- Identificar a variabilidade de uma LPS em artefatos de realização.
- Aplicar a variabilidade de uma LPS em artefatos de testes.

Introdução

A documentação e o gerenciamento da variabilidade são algumas das principais propriedades que identificam a engenharia de linha de produto de software (ELPS). A ELPS se distingue do desenvolvimento de um único sistema e da reutilização de software pela utilização de definições explícitas da variabilidade e da gestão dos seus aspectos.

Introduzir procedimentos de variabilidade no processo de desenvolvimento de software almeja principalmente facilitar a reutilização dos elementos de software, mapeando as variações que os componentes contêm como pontos concretos da arquitetura conforme as necessidades de cada projeto ou implementação. Quando recursos comuns e as variáveis dos aplicativos da LPS são identificados durante o subprocesso de gerenciamento de produtos, ocorre a introdução da variabilidade.

Os requisitos de domínio detalham os recursos definidos no gerenciamento de produtos, enquanto a variabilidade é carregada para esses requisitos. Tal abordagem é utilizada no projeto, na implementação e no teste.

Como essas engenharias lidam com modelos de um sistema em diferentes níveis de abstração, além da variabilidade do nível anterior a ser refinada, ajustando-se ao nível seguinte, uma nova variabilidade, condizente com o nível considerado e não consequente do refinamento do nível anterior, é adicionada. Por exemplo, a variabilidade no nível de implementação pode considerar diferentes tipos de linguagens de programação, algo que não era considerado no nível de projeto.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

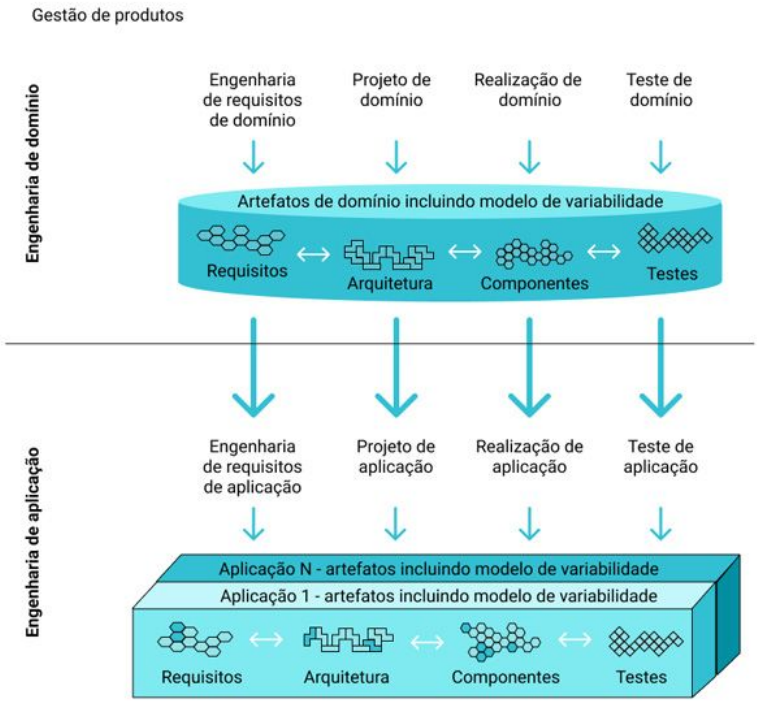
Definições gerais

Gerenciamento de variabilidade

O gerenciamento de variabilidade significa definir e explorar a variabilidade ao longo dos diferentes estágios do ciclo de vida de uma LPS. Esse gerenciamento engloba as seguintes questões:

- Atividades de apoio relacionadas à definição da variabilidade.
- Gerenciamento de artefatos variáveis.
- Atividades de apoio relacionadas à resolução da variabilidade.
- Coletar, armazenar e gerenciar as informações de rastreamento necessárias para cumprir essas tarefas.

Observe a imagem a seguir:



Foco da modelagem de variabilidade no framework da Engenharia de Linha de Produtos de Software (ELPS).

Todo subprocesso da engenharia de aplicação deve relacionar a variabilidade introduzida pelos subprocessos correspondentes da engenharia de domínio (figura 1.1) a fim de refinar a variabilidade encontrada em fases mais iniciais do desenvolvimento. Essa correlação tem de ser realizada de forma bastante segura para garantir que os requisitos variáveis sejam corretamente construídos.

Tal atividade é conhecida como **resolução de variabilidade**; o momento em que é executada, de **tempo de ligação da variabilidade**. Em nome da flexibilidade, esse tempo pode ser deixado para fases mais tardias do projeto.

Mundo real – sujeito e objeto de variabilidade

Na linguagem comum, variabilidade normalmente é relacionada à capacidade ou à propriedade de algo que favorece mudança. Na ELPS, a variabilidade não ocorre por acaso, sendo, ao contrário, provocada. No mundo real, ela pode se apresentar de diversas maneiras e em todas as coisas.

Por exemplo, uma lâmpada pode ser fluorescente ou de LED, um automóvel pode usar como combustível álcool ou gasolina ou uma lanterna pode utilizar pilhas ou baterias recarregáveis.

Dica

Com o intuito de perceber melhor o conceito de variabilidade, um bom exercício para você é tentar perceber no mundo real coisas que apresentem propriedades variáveis, como os exemplos citados.

Dois conceitos são utilizados para um melhor entendimento do significado da palavra “variabilidade”: sujeito da variabilidade e o objeto da variabilidade. Ambos serão apresentados a seguir.

Sujeito da variabilidade (SV)

Para um melhor entendimento do conceito de variabilidade, três perguntas podem ser feitas. A primeira é: **o que varia?** Tal pergunta leva diretamente ao conceito de SV.

O SV representa algum item do mundo real ou uma propriedade desse item que varia de alguma forma. São exemplos de SV:

As cores de uma casa

O tipo de material de uma porta

As formas de uma cadeira

A segunda pergunta é: **por que varia?** As variações podem se dever a uma série de condições.

Exemplo

Devido ao gosto do cliente, ao clima da região ou às leis do país.

No caso de SVs interdependentes, é possível inclusive que um item varie em decorrência da variação de outro item.

Exemplo

As cores de uma porta vão depender do tipo do material dela, porque uma porta de madeira pode ter determinadas cores que não se aplicam a uma de alumínio.

A terceira pergunta é: **como varia?** E essa pergunta nos leva ao conceito de OV.

Objeto da variabilidade (OV)

Ao pensarmos nos exemplos acima, imediatamente surgem as possibilidades de variações desses sujeitos de variabilidade.

Exemplo

As cores de uma casa podem ser verde, amarela ou azul; o tipo de material de uma porta, de madeira, ferro ou alumínio. Já as formas de uma cadeira podem variar entre o clássico e o moderno.

Essas possibilidades de variações são as **instâncias** possíveis de variações, sendo denominadas de OV.

Considerar as três questões acima causa um efeito importante na maneira de se pensar sobre a variabilidade. Ficar consciente da variabilidade e tratar dos seus aspectos subjetivos ou objetivos é um importante pré-requisito para que se possa realizar a modelagem de variabilidade.

Para fechar o entendimento de SV e OV, daremos mais dois exemplos:

SV “método de pagamento”

Observe que isso é algo do mundo real e que tem alguma variabilidade. O OV relacionado a tal sujeito são as possíveis formas de pagamento, que podem ser “em dinheiro”, “cartão de crédito” e “cartão de débito”.

SV "sistema de segurança"

Como SV, é possível haver um “mecanismo de identificação”, o qual, por sua vez, tem como OV as possíveis implementações do mecanismo, que podem ser “por impressão digital”, “por reconhecimento de voz” ou “por digitação de senha”.

É importante observar que diferentes OV podem influenciar objetos de variabilidade de outros itens. Nesse caso, a mudança do tipo de mecanismo de identificação (senha ou voz, por exemplo) influencia no tipo de armazenagem do banco de dados a ser utilizado (no caso de senha, um grupo de símbolos; no caso de voz, algo mais complexo).

Abstrações do mundo real

Abstrações do mundo real: ponto de variaçäo (ou de variabilidade) e variante

A seguir, assista ao vídeo e confira os principais pontos sobre este tópico.

> Conteúdo interativo
>
> Acesse a versão digital para assistir ao vídeo.

Ponto de variação ou de variabilidade (PV)

Como já compreendemos os conceitos de SV e OV, veremos agora como eles são incorporados na ELPS.

> Um PV é o correspondente na ELPS ao conceito de SV do mundo real. Essa definição se aplica a todos os tipos de artefatos do desenvolvimento, desde os requisitos até os testes, passando pela arquitetura, pelo projeto e pelo código.

Além de representar o SV incorporado, um PV possui informações contextuais sobre os detalhes da absorção da variabilidade do mundo real para o mundo do desenvolvimento de software, como o motivo de determinada variação ter sido incorporada.

Dessa forma, um PV é a representação de um SV enriquecido de informações contextuais.

Variante (V)

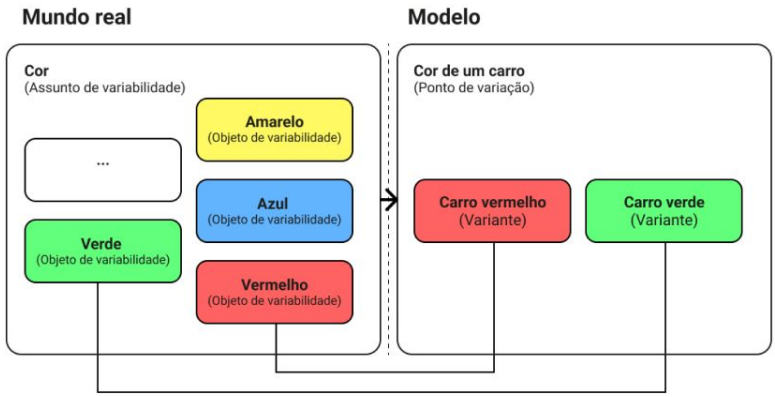
Enquanto um PV corresponde ao conceito de SV do mundo real, uma V corresponde à conceituação de OV do mundo real. Essa definição também se aplica a todos os tipos de artefatos do desenvolvimento, desde os requisitos até os testes. Uma V identifica uma única opção de um PV.

> PV e V são entidades independentes, distintas de artefatos de requisitos, de arquitetura, de tecnologia etc. Uma V pode ser associada a outros artefatos para indicar que eles correspondem a determinada opção da mesma forma que essa V corresponde a somente uma opção do PV correspondente.

Porém, nos casos em que não há perigo de confusão, os artefatos associados a uma V também são chamados de V.

Como exemplo de mapeamento de SV para PV e de OV para V, utilizaremos o SV “cor”. Na figura a seguir, estão associados a esse SV os objetos de variabilidade cores, os quais podem ser muitos: verde, branco, amarelo, vermelho, azul, marrom etc.

Quando pensamos em uma indústria automobilística, imaginamos que os carros produzidos possam ser de diferentes cores. Supondo que tal indústria deseje somente fabricar carros com as cores vermelho e verde, veremos a seguinte correspondência entre os conceitos:



Relação entre variabilidade no mundo real e um modelo do mundo real.

Definindo pontos de variaçãao (ou de variabillidade) e variantes

A definição de PVs e de Vs deve ser feita de maneira sistemática, buscando evitar a definição equivocada. Por isso, apresentaremos adiante um processo constituído de três passos para a realização dessa tarefa.

Primeiro passo

Identificar o item do mundo real que varia, o SV. Vamos usar como exemplo o desenvolvimento de um software de automação de residências.

Os engenheiros podem identificar diferentes maneiras de comunicação entre os componentes do software: LAN cabeada, LAN sem fio, Bluetooth etc. O que eles fizeram, nesse caso, foi identificar o SV “tipo de rede”.

Segundo passo

Definir o PV dentro da linha de produto de software. Trata-se de uma etapa necessária, porque há uma diferença entre as variações do mundo real (SV) e suas correspondentes na LPS (PV).

No caso do software de automação de residências, o SV “tipo de rede” identificado no primeiro passo resulta no PV “sistema de comunicação do software de automação residencial”. Esse PV, por sua vez, indica que o sistema deverá oferecer suporte para múltiplas formas de comunicação entre os componentes, sem, contudo, identificar quais.

Terceiro passo

Definir as Vs. Essa definição é baseada nos objetos de variabilidade do mundo real correspondentes ao SV inicial.

No caso do software de automação de residências, isso pode ser resolvido por questões técnicas e mercadológicas em que somente interessam dois tipos de comunicação: LAN sem fio e Bluetooth. Essas duas opções seriam as Vs do PV “sistema de comunicação do software de automação residencial” (definido no segundo passo).

Variabilidade em linha de produto de software

A seguir, assista ao vídeo e confira os principais pontos sobre este tópico.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Variabilidade e semelhança

A seguir, confira as principais diferenças entre variabilidade e semelhança:

Variabilidade

É a variação modelada para habilitar o desenvolvimento de aplicações customizadas pelo reuso de artefatos predefinidos e ajustáveis. Ela distingue as diferentes aplicações dessa linha de produtos.

Semelhança

É a característica de cada aplicação de uma linha de produtos implementada da mesma forma. É possível decidir se uma característica será variável ou se será comum a todas as aplicações da linha.

Por exemplo, no sistema de automação residencial, o usuário pode escolher a linguagem utilizada antes da instalação, o que caracteriza uma variabilidade. Por um valor adicional, também existe a possibilidade de ele escolher a linguagem a qualquer momento, mesmo depois da instalação.

Se o recurso de escolha da linguagem já é oferecido para todos os usuários na interface de usuário, isso significa que ele constitui um recurso comum a todos os aplicativos dessa linha, o que caracteriza uma semelhança.

Variabilidade no tempo versus variabilidade no espaço

Variabilidade no tempo

É o nome que se dá a uma variação de versão de um artefato de um sistema que ocorre ao longo do tempo. É inevitável que os sistemas evoluam com o decorrer do tempo, por exemplo, devido à evolução tecnológica.

> Tal variabilidade é aplicável tanto na engenharia de sistemas de software tradicional (sistemas únicos) quanto na ELPS. A técnica normalmente usada para gerenciar a variabilidade no tempo (evolução) é o gerenciamento de configuração.

Existe uma diferença significativa entre sistemas simples e sistemas de linhas de produto no que diz respeito à variabilidade no tempo. Como os softwares de linhas de produto possuem pontos de variação predefinidos, se torna mais fácil introduzir uma alteração evolutiva caso essa evolução venha a ocorrer em um desses pontos.

Exemplo

Em um sistema de automação residencial, se os engenheiros percebem que o modo de autenticação é capaz de mudar devido a uma evolução tecnológica, eles podem criar um PV “mecanismo de identificação” e definir uma única variação “cartão magnético”, desenvolvendo a linha de produto de forma a possibilitar futuras versões desse mecanismo. Isso tornará o sistema mais preparado para uma eventual variabilidade no tempo quando a V “leitor digital” estiver tecnologicamente madura para ser introduzida.

Essa “linguagem” para entender o comportamento do sistema não é disponibilizada em um desenvolvimento tradicional.

Variabilidade no espaço

É a existência de diferentes versões de um artefato de um sistema ao mesmo tempo.

Comentário

A engenharia de sistemas tradicional não provê uma maneira de lidar com a variabilidade no espaço. Já o objetivo da ELPS é construir produtos similares; normalmente oferecidos ao mesmo tempo, eles diferem em um escopo definido.

Entender e manipular a variabilidade no espaço é uma importante questão tratada pela engenharia de LPS para atingir esse objetivo.

Variabilidade interna e externa

Os clientes podem perceber as necessidades de variabilidade em um produto de maneira individualizada, percebendo apenas aquelas que satisfaçam às suas necessidades de usuário. Por outro lado, muitas variabilidades “escondidas” dos clientes são percebidas pelos engenheiros de LPS.

Essas duas possibilidades de perceber a variabilidade (visível ou não para os cliente) dão origem à classificação de variabilidade quanto à sua visibilidade em:

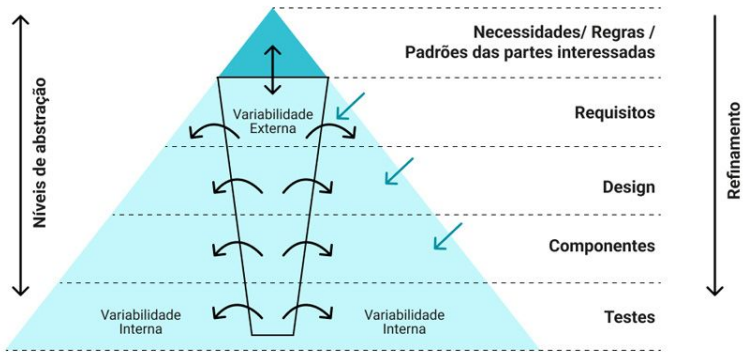
Variabilidade externa

É aquela que o cliente percebe e tem o poder de escolher as Vs de que necessitam.

Exemplo

O caso da escolha da maneira de identificação em um sistema de automação residencial: digitação de senha, cartão magnético ou leitor de digital.

Observe esta imagem:



Variabilidade em diferentes níveis de abstração.

Variabilidade interna

Podemos citar como exemplo a V “protocolo de comunicação” do mesmo sistema de automação residencial. A partir de medições da qualidade da rede, os instaladores podem decidir entre o protocolo que privilegia a

velocidade de transferência de dados e o que privilegia a correção dos dados. Essa decisão é transparente para o usuário.

É fácil perceber que, quanto mais baixo for o nível de abstração, desde as necessidades do usuário até os testes, passando pelos requisitos, pelo projeto e pela implementação de componentes, maior será a quantidade de variabilidade interna e menor a de variabilidade externa, como demonstra a imagem “Quantidade de variabilidade em diferentes níveis de abstração”.

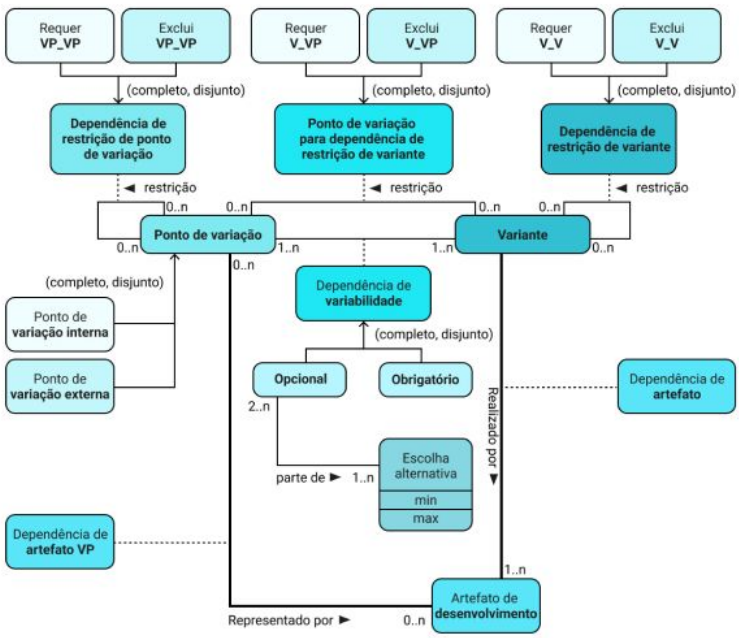
Modelo ortogonal de variabilidade (MOV)

Dependência de variabilidade

A documentação da variabilidade deve responder às seguintes perguntas:

- O que varia? Quais são os PVs?
- Por que varia?
  Causas internas (por exemplo, tecnologia);
  Causas externas (necessidades dos diferentes stakeholders).
- Como varia? Quais Vs estão relacionadas aos PV? Quais links há entre Vs e entre PVs? A quais partes do modelo se referem?
- Para quem é documentado? Exemplo: usuário (variabilidade externa) e engenheiro de testes (variabilidade externa).

MOV é uma linguagem para definir a variabilidade de uma linha de produto de software de forma ortogonal, proporcionando uma visão transversal da variabilidade para todos os artefatos da linha de produtos. São definidas notações gráficas para cada elemento do modelo por meio do uso de um diagrama de classes (notação UML 2). Observe a imagem a seguir:



Meta modelo de variabilidade.

No meio da mesma figura, os dois principais integrantes são o PV e a V. As classes que os representam se relacionam (1..n, 1..n), o que equivale a dizer que cada PV pode se relacionar com uma ou n Vs – e vice-versa.

O PV é uma classe abstrata que se especializa (completa e disjuntamente) em interno e externo, como já vimos. O relacionamento apresenta uma classe associativa abstrata denominada dependência de variabilidade, que se especializa da mesma forma em duas classes:

1.

Dependência de variabilidade opcional

2.

Mandatória

O significado dessa associação é que, em (1), a V de um PV pode ou não estar disponibilizada na aplicação, enquanto em (2), se o PV estiver presente, a V também deverá estar.

O exemplo de (1) é o caso do PV “mecanismo de autenticação”, que oferece as Vs “teclado”, “leitor de digital” e “cartão magnético”. O engenheiro pode escolher qualquer combinação, desde ter as três opções disponíveis até não ter nenhuma. A classe “Escolha alternativa”, que possui os atributos **min** e **max** e se relaciona com a classe “dependência de variabilidade opcional” (2..n, 0..1), restringe essa escolha.

Exemplo

Dadas as três Vs acima, é possível definir, por meio do relacionamento delas com a “escolha alternativa”, que somente duas podem ser escolhidas uma vez ou que as três podem sê-lo ao mesmo tempo, duas a duas, ou uma somente.

Já um exemplo de (2) seriam as mesmas Vs estarem presentes no produto ao mesmo tempo, não sendo possível ao engenheiro configurar que apenas uma delas fosse possível.

Restrição de variabilidade

Outra informação fundamental (parte superior da imagem “Meta do modelo de variabilidade”) diz respeito ao relacionamento entre:

- Vs de diferentes PVs; e
- Vs e PVs para Vs associadas a outro PV

Representados na parte de cima da imagem Meta do modelo de variabilidade, tais relacionamentos entre Vs e PVs (todos completos e disjuntos) podem ser de requisição ou de exclusão.

A semântica é que:

- Uma V pode requisitar ou excluir outra V do mesmo ou de outro PV (i – relacionamento de dependência de restrição de variantes).
- Uma V pode requisitar ou excluir qualquer V de um PV (ii – relacionamento de dependência de restrição de variante para ponto de variação).
- Um PV pode requisitar ou excluir as Vs de outro PV (iii – relacionamento de dependência de restrição de pontos de variação).

Como exemplo de (i), a V “WLan” (PV = “comunicação sem fio”) pode excluir a possibilidade da V “sensor baseado em radar” (PV = “detector de movimento”) ser utilizada, pois ambas são capazes de acessar a mesma faixa de banda.

Rastreabilidade entre variabilidade e artefatos

Não basta apenas modelar PV, V e seus relacionamentos. As conexões entre PVs e Vs e artefatos da LPS também são necessárias (parte inferior da imagem “Meta do modelo de variabilidade”) com o objetivo de documentar a rastreabilidade de Vs e PVs nos artefatos especificados em outros modelos (requisitos, projeto, implementação e testes).

Essa representação é obtida pela classe “Artefato de desenvolvimento” e por seus relacionamentos (1) com as classes V (“realizado por” – classe associativa “Dependência de artefato”) e VP (“representado por” – classe associativa “VP artefato”).

As multiplicidades desses relacionamentos indicam que:

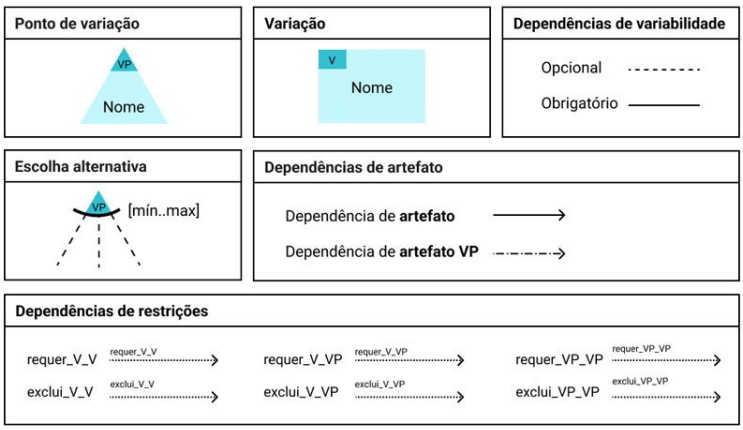
1. Um artefato pode ou não realizar uma ou mais Vs, mas toda V deve ser realizada por pelo menos um artefato;
2. Um artefato pode ou não representar um ou mais PVs e que cada PV pode ser realizado por nenhum ou muitos artefatos.

A representação de um PV por um artefato ocorre, por exemplo, quando o engenheiro quer representar uma classe abstrata que realiza o comportamento comum de várias classes V. Essa classe abstrata se relaciona com o PV.

Notação para uso do MOV

Para representar a informação de variabilidade definida a partir do metamodelo da imagem “Meta do modelo de variabilidade”, uma notação gráfica é utilizada, evitando, assim, a confusão de conceitos com os artefatos representados em outros modelos.

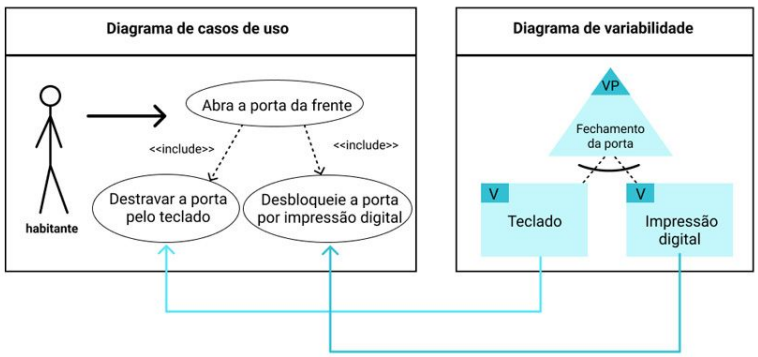
A imagem a seguir apresenta a notação:



Notação gráfica para modelo de variabilidade.

Um pequeno exemplo elucidará essa notação. Na imagem a seguir, o caso de uso “Abrir porta da frente” inclui dois outros: “Destravar porta pelo teclado” e “Destravar porta com digital”.

O modelo de variabilidade define o PV “Trava de porta” com duas Vs, tendo como escolhas alternativas default (1..1 ⇒ uma e somente uma pode ser escolhida): “teclado” e “digital”. Cada V é relacionada ao caso de uso respectivo por meio de uma dependência de artefato.



Rastreabilidade entre artefato e variantes.

Modelo ortogonal de variabilidade (MOV)

Assista ao vídeo a seguir com os principais pontos sobre o modelo ortogonal de variabilidade (MOV).

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Verificando o aprendizado

Questão 1

A engenharia de linha de produtos de software tem como uma das suas preocupações o desenvolvimento de artefatos de desenvolvimento que possam ser reutilizados em diversas aplicações de uma mesma linha de produtos. Para isso, um dos conceitos mais importantes é o de variabilidade, que diz respeito ao que varia em relação aos artefatos das aplicações. A variabilidade do mundo real é reconhecida por meio dos conceitos de

A

sujeito de variabilidade, que representa o que varia no mundo real, e objeto de variabilidade, o qual representa como o sujeito varia no mundo real.

B

objeto de variabilidade, que representa o que varia no mundo real, e sujeito de variabilidade, o qual representa como o objeto varia no mundo real.

C

sujeito de variabilidade, que representa o que varia no mundo real, e variante, o qual representa como o sujeito varia no mundo real.

D

ponto de variação (PV), que representa o que varia no mundo real, e objeto de variabilidade, o qual representa como o PV varia no mundo real.

E

ponto de variação (PV), que representa o que varia no mundo real, e variante, o qual representa como o PV varia no mundo real.

A alternativa A está correta.

A variabilidade do mundo real, muitas vezes refletida nos modelos que o representam, deve ser bem captada e entendida. Os objetos de variabilidade dizem respeito a tudo aquilo que pode variar, como, por exemplo, as cores de uma parede. Como podem variar, elas configuram outro aspecto que precisa ser identificado, o que é realizado pelo objeto de variabilidade. Por exemplo, os objetos de variabilidade do sujeito de variabilidade "cor da parede" podem ser as cores "verde", "amarelo" e "azul".

Questão 2

A variabilidade em uma linha de produtos de software representada por pontos de variação e por variantes pode se referir a mudanças decorrentes de processos evolutivos refletidos em requisitos funcionais ou não funcionais ou a variações de um mesmo artefato, que pode se apresentar de mais de uma maneira em diferentes aplicações dos produtos da linha. Em relação ao que foi dito, é correto afirmar que

A

uma variação que decorra de uma evolução ocorrida com a passagem do tempo é denominada variabilidade de espaço.

B

produtos similares com partes variáveis significam a existência de uma variabilidade no tempo.

C

a variabilidade no tempo representa, por exemplo, uma evolução tecnológica ocorrida e que faz com que uma nova versão do produto seja implementada.

D

os mesmos artefatos com versões diferentes em produtos similares lançados ao mesmo tempo são exemplos de variabilidade no tempo.

E

a engenharia tradicional possui mecanismos para definir claramente tanto a variabilidade no tempo quanto a variabilidade no espaço.

A alternativa C está correta.

A variabilidade no tempo diz respeito a variações ocorridas entre versões de um artefato que não coexistem ao mesmo tempo, pois derivam de evoluções que ocorrem com o passar do tempo, como, por exemplo, uma evolução tecnológica ou a mudança de uma lei. Já a variabilidade no espaço diz respeito a opções de variantes que podem ser implementadas ao mesmo tempo em diferentes aplicações da mesma

linha de produtos de software. Como exemplo, podemos citar a utilização de mais de um mecanismo de identificação de usuário em um sistema.

Variabilidade em artefatos de requisitos - artefatos textuais

A seguir, assista ao vídeo e confira os principais pontos sobre este tópico.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Variabilidade em artefatos de requisitos

Os requisitos de domínio abrangem requisitos comuns a todas as aplicações da LPS, bem como requisitos variáveis que permitem a criação de diferentes aplicações. O subprocesso de engenharia de tais requisitos é o momento de criação dos artefatos de requisitos de domínio.

Esses artefatos (requisitos textuais, casos de uso, modelo de dados etc.) são a entrada para o subprocesso de design de domínio, que se preocupa com o desenvolvimento da arquitetura do domínio. Por requisitos, entendemos:

- Uma condição ou necessidade do usuário para resolver um problema ou alcançar um objetivo.
- Uma característica que deve ser absorvida pelo sistema ou algum componente que satisfaça a um contrato, padrão ou outro documento formalmente exigido.

Requisitos podem ser documentados em linguagem natural ou em linguagem de modelagem de requisitos. A natural oferece mais flexibilidade; contudo, pode acarretar interpretações ambíguas. Já a de modelagem, com suas regras e elementos, tornam inequívoco o entendimento do significado da representação.

Existem **três** tipos de modelos de requisitos:

Requisitos funcionais

Temos como exemplo do primeiro requisito os diagramas de casos de uso.

Requisitos de dados

Temos como exemplo do segundo requisito o digrama de classe.

Requisitos de comportamento

Temos como exemplo do terceiro requisito o diagrama de estados.

Enquanto a meta é uma intenção que stakeholders anseiam alcançar com o sistema, uma funcionalidade se trata de uma característica visível ao usuário final. Metas e funcionalidades têm uma grande interseção.

Saiba mais

As metas são definidas pela equipe de engenharia de requisitos para expressar intenções de alto nível de abstração, enquanto as funcionalidades são modeladas pela equipe de projeto para expressar os requisitos de alto nível em características da arquitetura do sistema.

Variabilidade em requisitos textuais

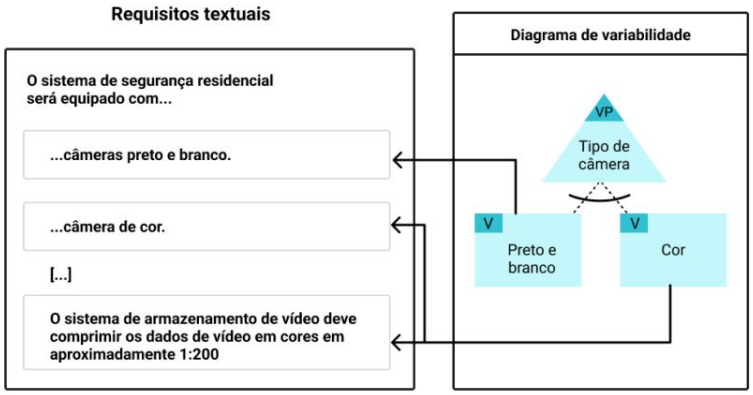
Apresentaremos agora um exemplo de documentação de variabilidade em requisitos textuais utilizando o MOV apresentado anteriormente. Esse exemplo evidenciará a importância de não transferir a ambiguidade do requisito textual variável para a modelagem da variabilidade.

> Imaginemos este requisito textual: “O sistema de segurança deve ser equipado com um sistema de câmeras ou em preto e branco ou coloridas capazes de capturar fotos infravermelhas. Se forem câmeras coloridas, o sistema de armazenamento deverá comprimir dados de vídeo a uma taxa de aproximadamente 1:200”. Uma ambiguidade é que não fica claro se a câmera em preto e branco deve capturar em infravermelho.

A identificação do PV e das Vs é mostrada a seguir:

Variabilidade em requisitos textuais.

Agora, observe sobre o MOV na imagem a seguir:

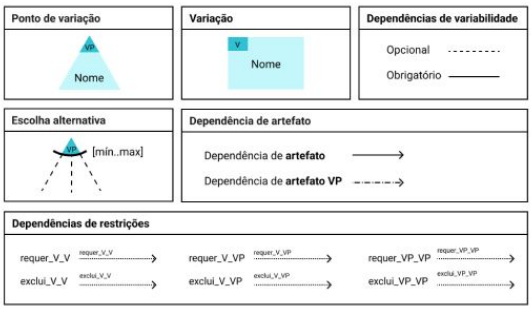


MOV em requisitos textuais.

Observe que o diagrama de variabilidade apresenta um PV (tipo de câmera) e duas Vs (preto e branco e colorido) como escolhas alternativas com cardinalidade default omitida (uma e somente uma pode ser escolhida). Duas dependências de artefato são definidas: a primeira, entre a V "preto e branco" e o artefato de texto "câmera em preto e branco"; a segunda, entre a V "colorida" e o artefato de texto "câmera colorida".

Observe que a segunda dependência de artefato se ramifica, adicionando mais uma dependência da V "colorida" com o artefato de texto que define a compressão dos dados de vídeo. Nesse caso, o MOV permitiu a seleção de um pedaço de texto correspondente à variante selecionada.

Qualquer dúvida quanto à notação utilizada nesse MOV pode ser sanada como já foi apresentado na imagem **Notação gráfica para modelo de variabilidade**. O texto pode ser aumentado de diferentes maneiras para melhorar a documentação da variabilidade.



Notação gráfica para modelo de variabilidade

Exemplo

Estruturas tabulares, diferentes tipos de estruturas de marcação ou hiper-referências.

Um emprego comum para a definição de artefatos de forma textual é por meio do uso de XML. Além disso, a inclusão de tags opcionais pode ser utilizada para a representação de variabilidade. Esse assunto é muito específico, sendo comentado apenas pelo fato de ser conhecido.

Variabilidade em artefatos de requisitos – artefatos de modelos

Variabilidade em casos de uso

Os modelos de requisitos (casos de uso, cenários, modelos de classes etc.) não são preparados para documentar a variabilidade como é requerido pela ELPS. O MOV permite que ela seja convenientemente documentada por meio de diferentes modelos sem alterar a forma original deles, o que veremos nesta seção. Por isso, apresentaremos aqui alguns exemplos de utilização do MOV com os modelos de casos de uso, diagrama de classes e diagrama de estados.

Um caso de uso é tipicamente documentado mediante o uso de:

- Uma descrição;

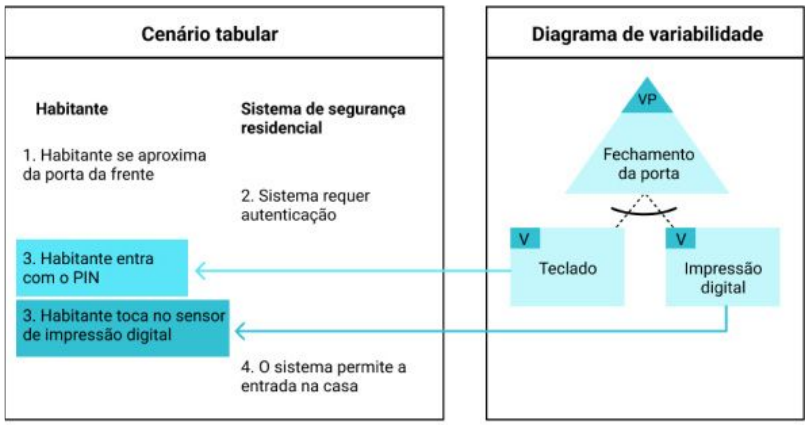
- Um cenário usando texto tabulado ou diagrama de sequência;
- Um diagrama de caso de uso.

Caso os requisitos sejam variáveis, o MOV se adequará perfeitamente à função de documentar a variabilidade, ligando-se aos modelos de artefatos citados, conforme veremos a seguir.

As imagens a seguir apresentam dois modelos (tabular e diagrama de sequência) que apresentam a mesma situação: passos básicos para fechar a porta da frente de uma casa inteligente. Fica evidente que esses passos contêm opções que dependem da V escolhida (teclado ou leitor de digital). Observe como o MOV se relaciona com os dois modelos, indicando como as Vs se ligam às partes dos artefatos, o que torna inequívoca a identificação de variabilidade nos modelos.

Conteúdo interativo

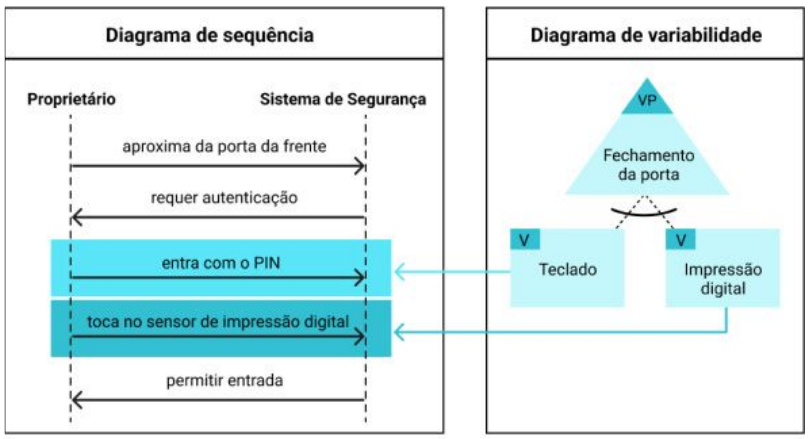
Acesse a versão digital para ver mais detalhes da imagem abaixo.



Exemplo de documentação de variabilidade em um cenário tabular.

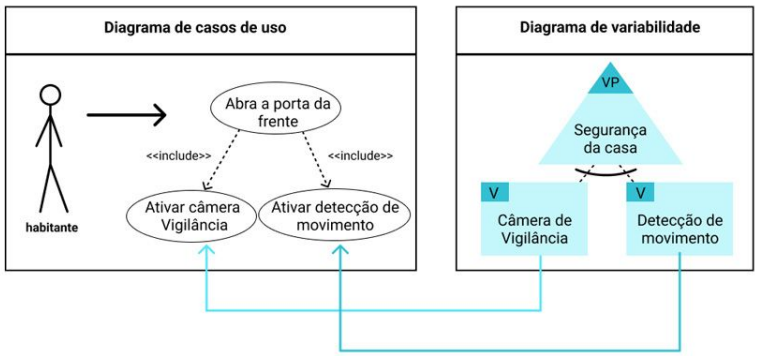
Conteúdo interativo

Acesse a versão digital para ver mais detalhes da imagem abaixo.



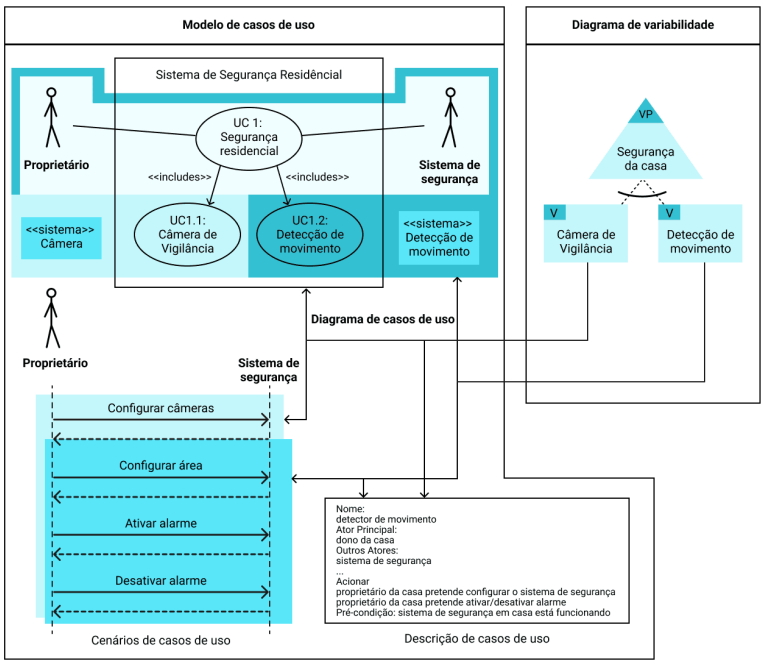
Exemplo de documentação de variabilidade em um cenário usando notação de diagrama de sequência.

Já a próxima imagem mostra a documentação de variabilidade em um diagrama de casos de uso. Na comparação com as notações vistas acima, tais diagramas apresentam os requisitos de um ponto de vista de mais alto nível.



Exemplo de documentação de variabilidade em um caso de uso.

Observe a imagem a seguir:



Exemplo de documentação de variabilidade em vários modelos.

É importante perceber que **o uso do MOV permite correlacionar as Vs presentes em diversos modelos**, o que pode ser muito difícil de ser percebido – talvez impossível, caso pensemos em grandes sistemas. A imagem já apresentada nos dá uma boa ideia dessa capacidade.

Variabilidade em diagrama de classes

Diagramas de classes documentam a variabilidade de requisitos estruturais. Essa variabilidade estrutural capturada em um diagrama de classes pode envolver:

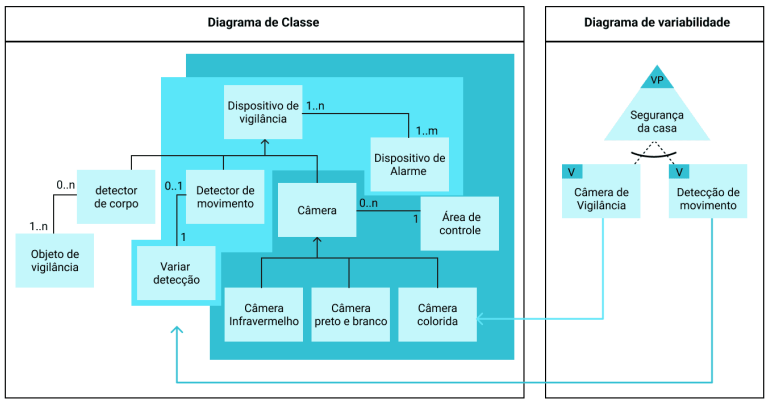
- Variabilidade de comportamento (associações).

- Função (métodos).
- Qualidade (atributos).

Essas variabilidades são documentadas em outros modelos de requisitos, como, por exemplo, casos de uso e diagramas de estado. A variabilidade é inerente ao modelo de classes devido ao fato de que esse modelo especifica um conjunto de instâncias (objetos e suas ligações), sendo usada para validar essas instâncias em tempo de execução.

Contudo, a variabilidade da linha de produto não pode ser documentada usando as notações padrão do modelo de classes. Para essa finalidade, mais uma vez o MOV é utilizado.

A imagem a seguir ilustra a utilização do MOV para a documentação da variabilidade em um modelo de classes. Observe que a variabilidade documentada apresenta uma granularidade “grossa”, ou seja, engloba classes e subclasses. Variabilidades com uma granularidade mais fina teriam uma ligação de dependência não com classes, e sim com elementos de classes, como, por exemplo, atributos, métodos e relacionamentos.

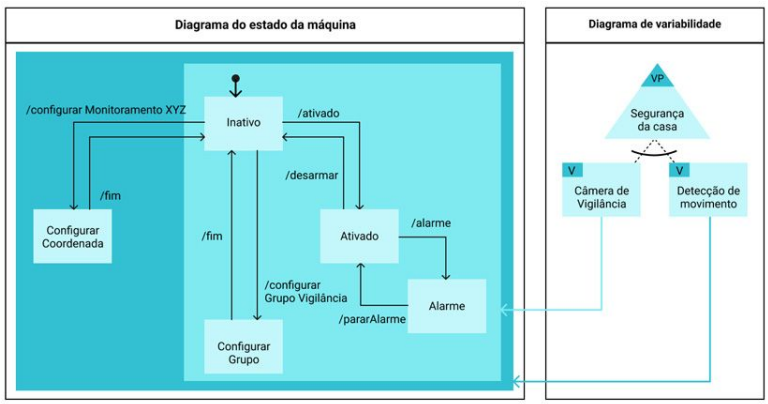


Exemplo de documentação de variabilidade em um diagrama de classes.

Variabilidade em diagrama de estados

O modelo de máquina de estados documenta requisitos relacionados ao comportamento da aplicação. A documentação da variabilidade ocorre da mesma forma utilizada na documentação dos demais artefatos de projeto: ligando as variantes a certas partes do diagrama de máquina de estados.

A imagem a seguir apresenta um exemplo de modelagem de comportamento de variantes que usa o MOV e um diagrama de estados, ambos devidamente conectados por relações de dependência de artefatos.



Exemplo de documentação de variabilidade em um diagrama de estados.

Variabilidade em artefatos de requisitos – artefatos de modelos

Assista ao vídeo a seguir com os principais pontos sobre variabilidade em artefatos de requisitos.

Conteúdo interativo

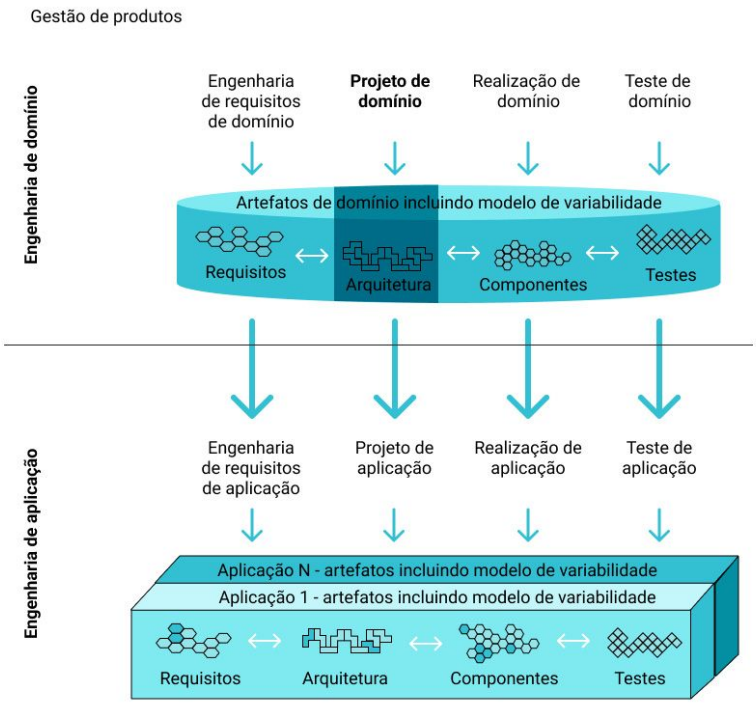
Acesse a versão digital para assistir ao vídeo.

Variabilidade em artefatos de projeto

Estrutura arquitetural

Os desejos do usuário devem ser satisfeitos nas aplicações, e a maneira de conseguir isso é implementar os requisitos. Na fase do projeto, tecnicamente falando, determina-se como as aplicações são construídas.

Na ELPS, a arquitetura do domínio (ou de referência) deve ser válida para várias aplicações da mesma linha, o que torna necessário que os artefatos de domínio (requisitos, projeto, implementação e testes) sejam flexíveis. Já o MOV permite a documentação da variabilidade de uma maneira clara e compreensível, tornando fácil o reuso dos artefatos produzidos. Por fim, o subprocesso de projeto do domínio é responsável pela criação dos artefatos de projeto comuns e variáveis de uma LPS, como aponta a imagem a seguir.



A arquitetura de referência é o foco da documentação de variabilidade no projeto.

O projeto do software engloba **duas fases**:

Projeto de alto nível

A arquitetura da aplicação é o principal produto dele e inclui a estrutura do software.

Projeto de baixo nível

É a realização (implementação – subprocesso de realização do domínio), que deve ser aderente à arquitetura.

Os engenheiros de arquitetura têm quatro preocupações:

1.

Identificar quais requisitos têm impacto essencial na arquitetura

2.

Criar uma arquitetura conceitual antes de construir os modelos de software

3.

Decompor o software em partes e relacionamentos (estrutura arquitetural)

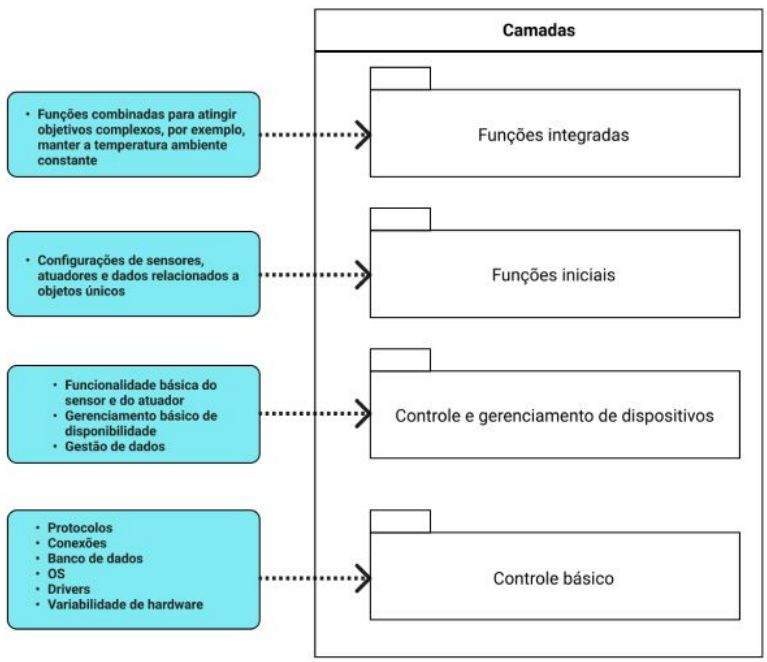
4.

Definir a coleção de regras comuns para a realização do sistema (textura arquitetural)

A imagem a seguir possui um exemplo de estrutura arquitetural em camadas para o sistema de automação residencial. A estrutura é representada pelas quatro camadas, cada uma com suas estruturas internas e componentes, e tendo funcionalidades bem definidas.

Conteúdo interativo

Acesse a versão digital para ver mais detalhes da imagem abaixo.



Exemplo de estrutura arquitetural em camadas para o sistema de automação residencial.

Como exemplo, a textura poderia ser representada pelo fato de que o sistema é organizado em camadas, usando os padrões de projeto Facade (proporciona uma interface simples entre os subsistemas) e Observer

(separa a interface do usuário dos dados), entre outros exemplos. Dessa forma, a textura lida com as regras de codificação e os mecanismos gerais, como estilos e padrões de projeto.

Visões da estrutura arquitetural

A estrutura arquitetural é normalmente constituída por visões, que, juntas, formam a estrutura completa. As principais visões são:

Lógica

Incorpora os requisitos do modelo, descrevendo as aplicações segundo o problema do domínio. Normalmente, ela é expressa em termos de artefatos de requisitos.

Desenvolvimento

Decompõe o software em componentes, objetos e interfaces. É composto por estes diagramas: pacote, componentes, classes e objetos.

Processo

Explicita as atividades durante a execução. Principais diagramas: estado, atividade e sequência.

Código

Mapeia os códigos-fonte e o código executável em arquivos e diretórios, distribuindo-os em unidades de processamento. Utiliza ainda os diagramas de implantação e de componentes.

As decisões sobre variabilidade tomadas durante o projeto precisam ser documentadas para uso futuro. Denominada arquitetura de referência, a representação clara de PV, de V e de mecanismos que realizam a variabilidade é obtida durante a engenharia de domínio.

Documentando a variabilidade – visão de desenvolvimento

Assista ao vídeo a seguir com os principais pontos sobre a documentação da variabilidade.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Apresentaremos neste estudo exemplos de documentação de variabilidade por meio do uso do MOV em conexão com artefatos de projeto, compondo as visões de desenvolvimento, de processo e de código.

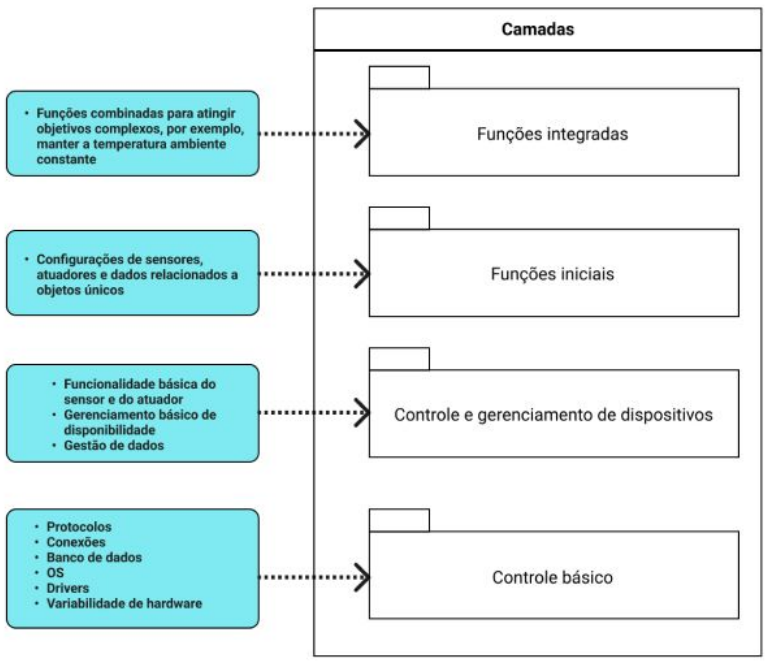
Comentário

A documentação da variabilidade na visão lógica já foi abordada anteriormente neste estudo.

Variabilidade na estrutura arquitetural

A visão de desenvolvimento é uma das mais importantes para a captura da variabilidade e da semelhança. Um subsistema é decomposto em uma coleção de componentes interativos.

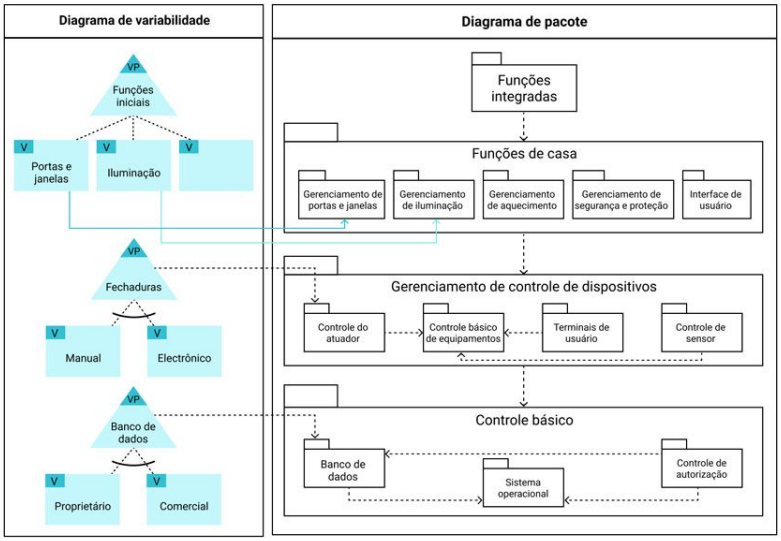
A estrutura arquitetural em camadas apresentada pela imagem **Exemplo de estrutura arquitetural em camadas para o sistema de automação residencial** pode ser refinada com a utilização de um diagrama de pacotes e ter o modelo de variabilidade incluso, como pode ser observado na imagem a seguir. Observe a relação entre os pontos de variabilidade e os elementos da arquitetura, o que indica que o artefato na extremidade alvo possui instâncias diferentes.



Exemplo de estrutura arquitetural em camadas para o sistema de automação residencial

Conteúdo interativo

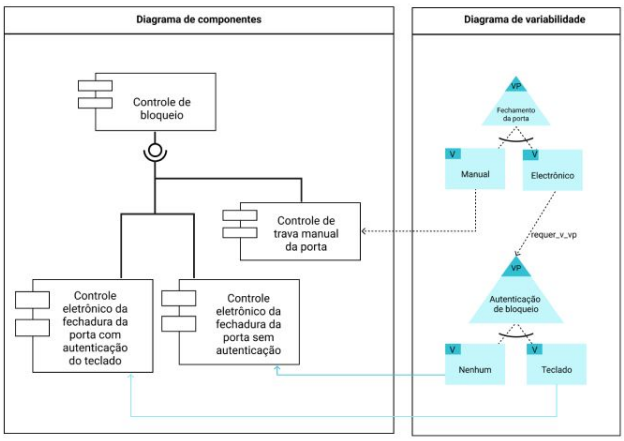
Acesse a versão digital para ver mais detalhes da imagem abaixo.



Variabilidade na estrutura arquitetural em camadas para o sistema de automação residencial.

Variabilidade nos componentes

A UML 2 proporciona o diagrama de componentes para descrever a configuração de todos os componentes do software. Apresentada na imagem seguir, a documentação da variabilidade com esse diagrama destaca a variabilidade em parte do componente “gerenciamento de portas e janelas” – subsistema controle de travamento de portas. Observe a imagem a seguir:



Variabilidade no componente “gerenciamento de portas e janelas”.

É possível notar que:

- Cada componente pode realizar uma V.
- A V “eletrônico” do PV “Travamento de porta” tem uma restrição de variabilidade em relação a outro PV, “Autenticação de travamento”.

Se tiver qualquer dúvida, reveja o conteúdo **Notação para uso do MOV**.

Os componentes são conectados entre si por meio de interfaces. Uma interface proporciona a descrição de funcionalidade de um componente provedor para um requisitante, deixando em aberto como a implementação da funcionalidade será realizada. Interfaces são implementadas por código, e cada linguagem possui um mecanismo para esse fim.

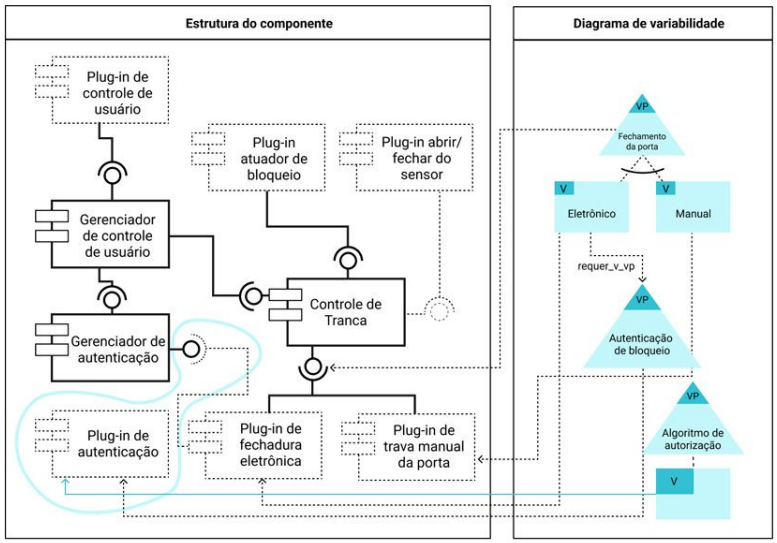
Na imagem apresentada, o componente “controle de travamento” disponibiliza uma interface utilizada pelos outros três componentes daquele pedaço de modelo.

Configuração de componentes

Documentar a variabilidade nos componentes implica configurar os componentes e as interfaces de forma a representar as possibilidades de variações. É preciso especificar quais configurações são possíveis e quais não o são. Isso define quais componentes são comuns e quais são opcionais ou variáveis.

Tais restrições podem ser feitas por meio de listas ou de um framework de componentes. O framework, que é um diagrama de componentes, contém componentes plugin que podem ser adicionados a localizações plugin, e essas adições obedecem a regras que mantêm a variabilidade dentro do requerido.

Focando o subsistema “controle de travamento de portas”, a imagem a seguir exibe o framework de parte de um sistema de automação de residências:



Framework componente para parte do sistema de automação residencial.

No framework, partes opcionais e variantes são conectadas com linhas tracejadas. Componentes plugin contêm essa expressão nos seus nomes, e as localizações plugin são interfaces requeridas. Os pontos de variação são ligados a interfaces; às suas variantes, a componentes que se conectam às mesmas interfaces, mantendo a consistência do modelo.

Observe a restrição de dependência de requisição (veja o conteúdo **Notação para uso do MOV**) existente entre a V “eletrônico” do PV “travamento de porta” e o PV “autenticação de travamento”. Essa restrição se reflete no diagrama de componentes. Um maior aprofundamento sobre o framework é feito na etapa de projeto da engenharia do domínio.

Documentando a variabilidade – visão de processo e visão de código

Variabilidade na visão de processo

A visão de processo descreve o comportamento da aplicação durante a execução em relação a threads (tarefas executadas), processos, suas interações e recursos usados. A variabilidade nessa visão se manifesta como:

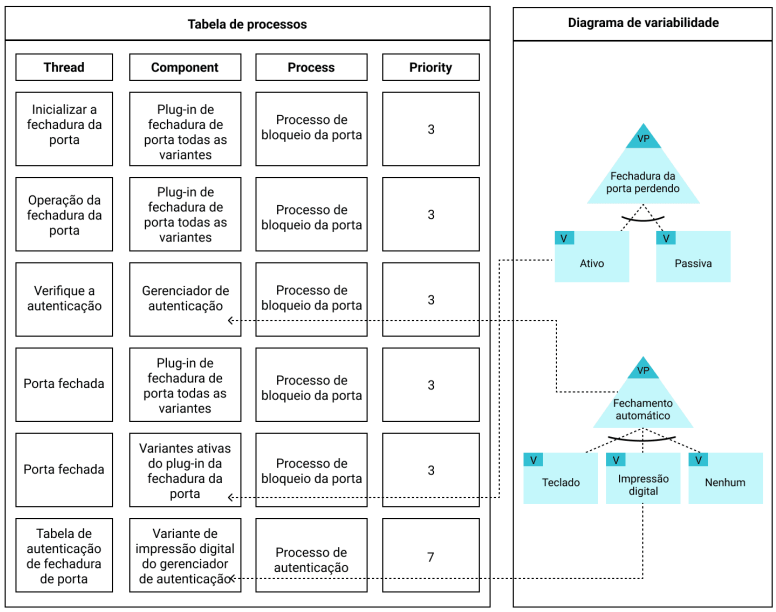
- Diferentes agrupamentos de tarefas.
- Tarefas e processos que podem ser opcionais e/ou ter múltiplas instâncias.
- Esquema de prioridades de processos.
- Mecanismos de comunicação entre processos.

A velocidade, o rendimento e o tempo de reação aos eventos dos aplicativos são afetados pelas escolhas. Na ELPS, a visão de processo pode ser afetada por requisitos variáveis que tratam de desempenho ou de outras

questões de qualidade e por uma variabilidade interna influenciada pelo hardware ou pela infraestrutura básica utilizada.

Os diagramas usados na visão do processo não possuem uma notação para variabilidade. A maioria desses diagramas apenas descreve semelhanças. Além de aplicar o MOV nos diagramas, é possível lançar mão de outros meios – e um deles é uma tabela de processos. Essa tabela relaciona processos, tarefas e componentes, como pode ser visto imagem a seguir.

Tarefas e componentes aos quais são atribuídas tarefas variam junto. Se o componente for opcional ou tiver várias instâncias, as atribuídas a eles vão variar da mesma maneira.



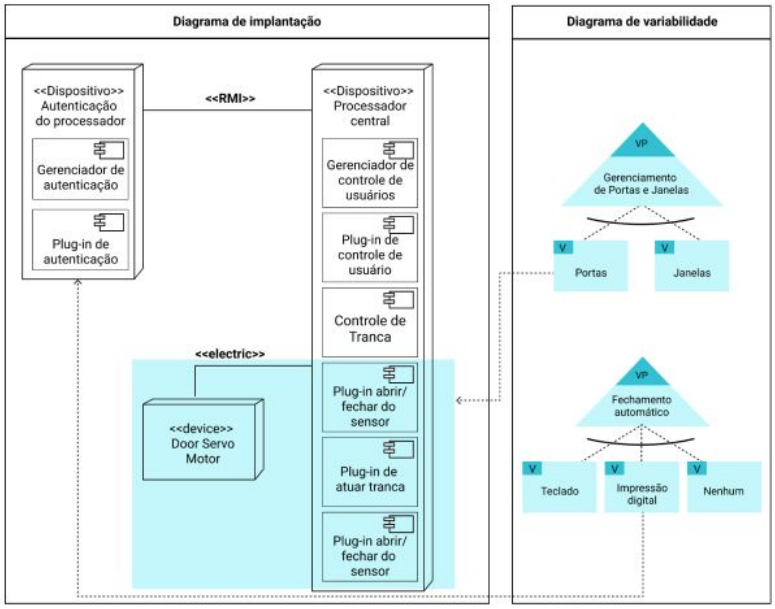
Exemplo de documentação de variabilidade na visão de processo.

Variabilidade na visão de código

A visão de código lida com a distribuição do código-fonte em arquivos e diretórios e com os arquivos executáveis rodando nas unidades de processamento. A variabilidade pode ocorrer em:

- Arquivos;
- Diretórios;
- Unidades de processamento;
- Número e papéis das unidades de processamento;
- Mapeamento do código nas unidades de processamento.

Os diagramas que documentam a visão de código são os de implantação e de componentes. A imagem a seguir apresenta a documentação de variabilidade na visão de um código utilizando um diagrama de implantação.



Exemplo de documentação de variabilidade na visão de código.

Verificando o aprendizado

Questão 1

Os requisitos de domínio, tanto aqueles que são comuns a todas as aplicações da LPS quanto os chamados variáveis, que permitem desenvolver aplicações distintas, podem ser documentados em uma linguagem natural ou de modelagem. Em relação à documentação de requisitos, é correto afirmar que

A

a documentação em linguagem natural oferece mais flexibilidade e mais clareza.

B

a documentação em linguagem de modelagem oferece mais flexibilidade; contudo, é mais ambígua.

C

a documentação em linguagem natural oferece menos flexibilidade e mais certeza.

D

o diagrama de classes pode ser utilizado como um modelo de documentação de requisitos de dados.

E

existem três tipos de modelos de requisitos: funcional, de dados e operacional.

A alternativa D está correta.

Os requisitos podem ser de três tipos: funcionais, de dados e de comportamento. O diagrama de classes pode ser usado como modelo para a documentação dos requisitos de dados, pois armazena a informação sobre a estrutura dos dados e seus relacionamentos.

Questão 2

A estrutura arquitetural é constituída de visões, e cada uma delas abrange um aspecto do modelo de requisitos. As visões são realizadas por meio do uso de diagramas adequados a cada uma; juntas, elas formam a estrutura completa. Em relação às visões da estrutura arquitetural, é correto afirmar que

A

a visão de código permite a documentação de atividades durante a execução.

B

a visão de desenvolvimento distribui os códigos-fonte e é executável por arquivos e diretórios.

C

a visão de processo permite entender os processos realizados pelas pessoas que utilizam o software.

D

a visão lógica e a de código são muito próximas entre si, havendo mesmo uma grande interseção entre esses modelos.

E

os diagramas de classe e de objeto são adequados para a documentação da visão de desenvolvimento.

A alternativa E está correta.

Os diagramas de classe se prestam a documentar tanto a visão lógica (dados) quanto a de desenvolvimento, sendo refinados ao incorporar classes introduzidas ao longo do processo de desenvolvimento e que não sejam de entidade. O diagrama de objeto mostra a estrutura de relacionamento das instâncias das classes em um momento de execução. Dessa forma, ambos servem para elucidar parte da visão de desenvolvimento, que é composta por componentes, objetos e interfaces.

Artefatos de projeto detalhado

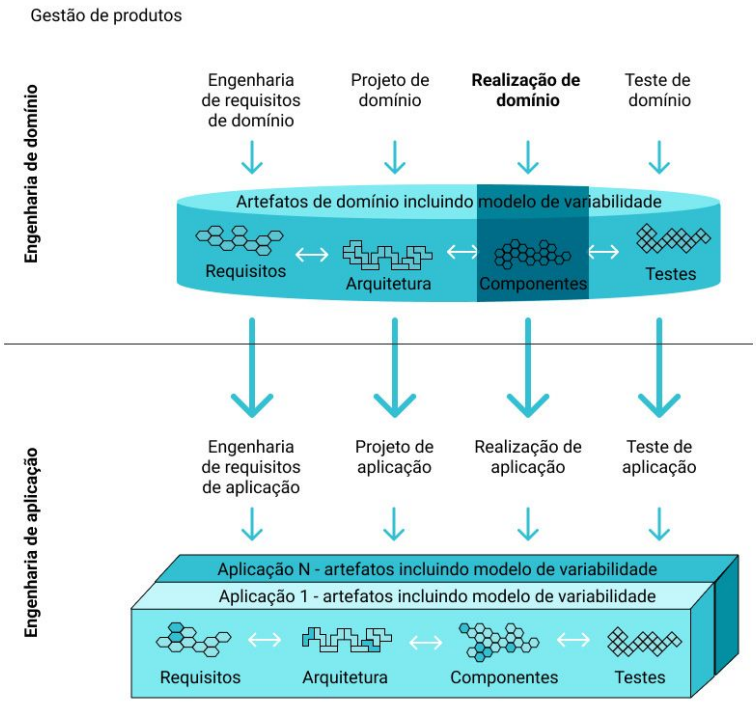
A seguir, assista ao vídeo e confira os principais pontos sobre este tópico.

> **Conteúdo interativo**
>
> Acesse a versão digital para assistir ao vídeo.

Existem muitas maneiras de implementar a variabilidade em linguagens de programação e ferramentas que suportam o desenvolvimento. Estudar essas maneiras, porém, não faz parte do escopo deste conteúdo.

O subprocesso de realização de domínio se concentra na criação de artefatos de realização comuns e variáveis, incluindo os componentes reutilizáveis, e tem como entrada os produtos do projeto de domínio (visto no módulo anterior), que são a estrutura arquitetônica e a textura arquitetônica.

A estrutura determina os componentes e as interfaces que devem ser projetados e implementados, documentando também a variabilidade externa e interna. Já a textura fornece diretrizes comuns que especificam regras comuns para lidar com a variabilidade no projeto e a implementação de componentes e interfaces. Observe a imagem a seguir:

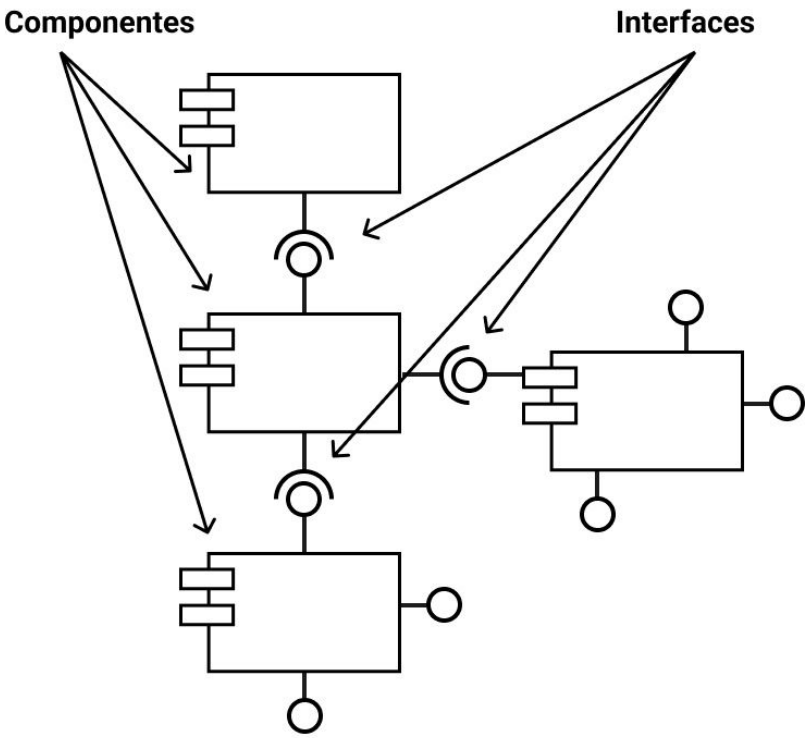


Foco da modelagem de variabilidade no subprocesso de realização do domínio.

O projeto detalhado lida com o projeto de componentes e interfaces, os quais, por sua vez, são determinados pela estrutura da arquitetura. No projeto do domínio, os componentes e as interfaces são definidos. Na realização do domínio, ambos são caracterizados, descendo um nível em relação às suas especificações. As imagens a seguir oferecem uma visão comparativa do alcançado em cada um desses subprocessos (projeto do domínio e realização do domínio).

Conteúdo interativo

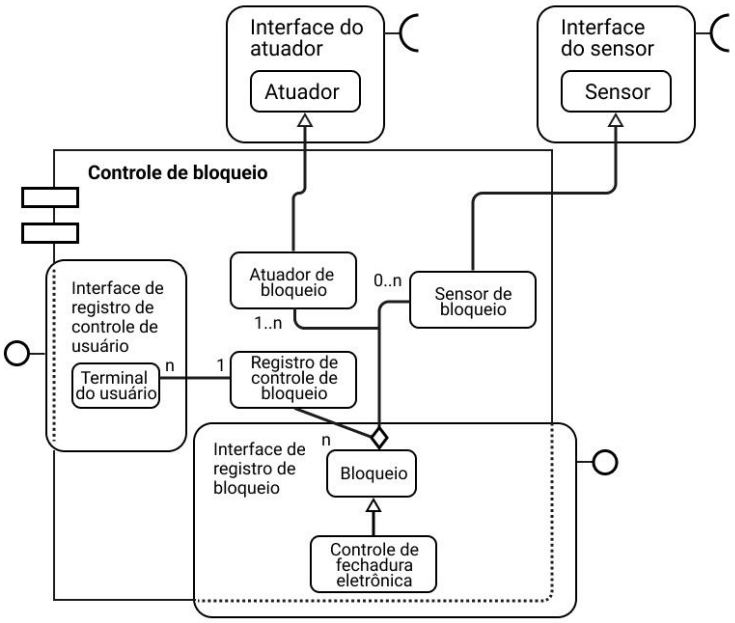
Acesse a versão digital para ver mais detalhes da imagem abaixo.



Componentes e interfaces no subprocesso de projeto do domínio.

Conteúdo interativo

Acesse a versão digital para ver mais detalhes da imagem abaixo.



Refinamento de componentes e interfaces no subprocesso de realização do domínio.

Enquanto componentes são as peças principais a partir das quais os aplicativos são construídos, as interfaces são as partes externamente visíveis dos componentes utilizadas para conectá-los. Eles fornecem funcionalidade a outros componentes por meio de uma interface fornecida, além de usarem uma interface necessária para acessar a funcionalidade fornecida por outros componentes.

Comentário

Se um componente fizer parte da plataforma, ele será usado em muitos aplicativos. Além disso, uma falha no seu desenvolvimento se espalhará por toda a LP. Dessa forma, o projetista de um componente ou interface precisa conhecer muito bem o lugar e o papel do componente ou interface na arquitetura.

Variabilidade da interface do componente

Interfaces são mecanismos importantes para implementar a variabilidade. A mesma interface pode ser fornecida por vários componentes.

Além disso, interfaces podem ser requisitadas por outros componentes. Isso resulta em um grande número de configurações possíveis de componentes de domínio em aplicativos de linha de produtos.

Cada combinação pode representar uma forma diferente de o aplicativo funcionar, caracterizando sua variabilidade. A restrição mais importante em uma interface é a variabilidade nos diferentes componentes que precisam ser conectados.

Esses tipos de variabilidade têm efeitos no projeto de uma interface:

O uso de diferentes algoritmos ou protocolos

Diferenças nos recursos fornecidos

Diferenças na configuração do aplicativo

Muitos componentes de fornecimento

Variabilidade em algoritmos ou protocolos

A mesma interface pode ser implementada de maneiras distintas e por diferentes métodos dos objetos das classes. Os tipos de argumentos e de retornos de tais métodos precisam ser escolhidos de tal forma que cada variação de algoritmo possa lidar com eles.

> Em linhas gerais, a interface define os tipos dos argumentos e os de retorno da funcionalidade, sem especificar os algoritmos. Por sua vez, os diversos componentes variáveis que fornecem essa interface precisam implementar os algoritmos (variáveis) cada um da sua maneira, mas sempre usando os argumentos e o retorno dos mesmos tipos definidos na interface.

Além disso, os componentes (variáveis) que utilizam a funcionalidade desses componentes via interface também devem estar preparados para enviar parâmetros e receber os argumentos dos mesmos tipos – por mais diferentes que sejam uns dos outros.

Variabilidade em recursos

Componentes diferentes lidam com o mesmo tipo de funcionalidade, embora possam fornecer ou usar quantidades diferentes de determinados recursos. Listaremos alguns exemplos de tais recursos:

- Tamanhos de memória;
- Tempo de processamento;
- Espaço de tela;
- Largura de banda;
- Velocidade de comunicação.

Em muitos casos, o componente que requer a interface precisa de pelo menos algumas informações abstratas sobre o tamanho do recurso necessário. Normalmente, o valor referente a uma escala pode ser mais ou menos abstrato, como, por exemplo, o tamanho absoluto da tela em pixels ou apenas uma distinção entre grande, médio e pequeno.

Pode ser necessário que a interface forneça funções ou parâmetros que indiquem o nível dos recursos atualmente disponíveis, permitindo que uma avaliação seja feita em tempo de execução pelo componente utilizador. Também é desejável que a interface forneça mecanismos para a realocação dos recursos.

Variabilidade na configuração da aplicação

Em muitos casos, a variabilidade está relacionada à configuração do aplicativo, incluindo diferenças de hardware e software, como, por exemplo, diferentes tamanhos de memória ou diferenças na disponibilidade de determinados pacotes de software.

> Os componentes obrigatórios precisam de uma visão abstrata da configuração. Consequentemente, a interface deve conter funções, métodos ou parâmetros que diferenciem as variantes.

As diferenças nos recursos podem ser vistas como um caso especial de diferenças na configuração. Enquanto os recursos estão relacionados às propriedades internas de hardware e software, a configuração está relacionada a todos os tipos de propriedades do sistema.

Componentes versus interface

Uma interface pode ser utilizada por vários componentes, e isso pode ocorrer em relação aos processos que afetam a maioria dos componentes, como, por exemplo, inicialização e tratamento de erros. A interface carrega apenas algumas classes de objetos que possuem poucos métodos, os quais, em geral, são muito genéricos.

Os métodos possíveis de serem chamados **via interface**, que não são implementados nela, podem ter diversos tipos de implementação segundo a parte do sistema que usa **aquela interface**. Obviamente, diferentes classes implementarão esses métodos de acordo com as partes do sistema que serão requisitantes da interface. Isso é uma forma de implementar a variância do sistema.

Variabilidade interna do componente

Assista ao vídeo a seguir com os principais pontos sobre a variabilidade interna do componente.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

As necessidades de variabilidade são atendidas na realização do componente de duas maneiras não excludentes:

1° Fornecer variantes diferentes de um mesmo componente

2° Dar variabilidade dentro do componente

Como os componentes são normalmente formados por várias classes, os diagramas de classe são usados para documentar sua estrutura interna. Essa estrutura consiste principalmente em classes de objetos que interagem. Algumas delas são definidas nas interfaces fornecidas e obrigatórias; outras, relacionadas à parte interna do componente.

O diagrama de classes da imagem a seguir representa um componente de controle de bloqueio. Note que ele possui várias interfaces fornecidas e requeridas.

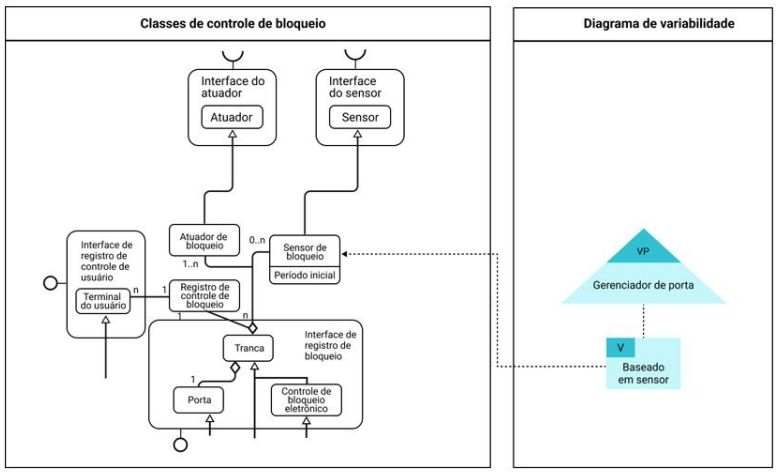


Diagrama de classes com documentação da variabilidade do plugin “controle de bloqueio”.

Na parte superior da imagem, estão representadas duas classes necessárias para o controle de atuadores e sensores. As classes dessas interfaces são usadas por herança para as classes “sensor de bloqueio” e “acionador de bloqueio” no componente de controle de bloqueio.

As interfaces requeridas têm suas funcionalidades implementadas nas classes que são filhas pelo mecanismo de herança. Tal mecanismo garante a documentação da variabilidade nesse ponto do modelo.

Atenção

Não faz parte do escopo deste conteúdo aprofundar o significado do diagrama de classes mostrado na imagem, e sim apresentar os conceitos de documentação de variabilidade utilizados.

Deve ficar clara a existência de diversas formas de descrever a variabilidade em diagramas de classe, como herança, anotações de multiplicidade e atributos de classe. A herança é usada com o intuito de fornecer variantes para classes abstratas disponíveis nas interfaces necessárias.

Boa parte da variabilidade no diagrama de classes é aquela relacionada às instâncias de tempo de execução do modelo de classe, o que não guarda relação com a variabilidade da linha de produtos, ou seja, a variância que diferencia um produto do outro.

Comentário

O mecanismo de herança é o item realmente importante para a engenharia de linha de produtos de software. Sua utilização é feita em conjunto com o modelo de variabilidade ortogonal para capturar a variabilidade da linha de produtos.

Deve-se tomar cuidado para determinar o que significa variabilidade durante o projeto detalhado, pois, como já frisamos, a variabilidade no diagrama de classes pode ser usada para definir diferentes instâncias de tempo de execução e para distinguir entre variantes.

O projetista pode recorrer a outros meios para distinguir claramente entre esses casos, como, por exemplo, o texto explicativo. O importante é que não fique dúvida sobre o conceito de uma variação devido ao tempo de execução e uma devido à variabilidade de uma LPS.

Variabilidade da interface do componente

A seguir, assista ao vídeo e confira os principais pontos sobre este tópico.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Verificando o aprendizado

Questão 1

O subprocesso de realização de domínio tem como entrada os produtos do projeto de domínio e como objetivo a criação de artefatos de realização comuns e variáveis, incluindo os componentes reutilizáveis. Em relação ao uso da estrutura e da textura arquitetônica dos produtos do projeto de domínio como entradas para a realização do domínio, é correto afirmar que

A

a estrutura determina os componentes e as interfaces que devem ser projetados e implementados, mas não a variabilidade externa e interna.

B

a estrutura fornece diretrizes comuns que especificam regras comuns para se lidar com a variabilidade no projeto e a implementação de componentes e interfaces.

C

a textura determina os componentes e as interfaces que devem ser projetados e implementados, mas não a variabilidade externa e interna.

D

a textura fornece diretrizes e regras comuns para lidar com a variabilidade somente no projeto.

E

a estrutura determina os componentes e as interfaces que devem ser projetados e implementados, documentando também a variabilidade externa e interna.

A alternativa E está correta.

Como produtos do subprocesso de projeto de domínio, a estrutura e a textura têm como foco respectivamente a determinação de componentes e interfaces a serem projetados e implementados (com sua variabilidade externa e interna) e a definição de regras para implementar a variabilidade em todas as fases do desenvolvimento.

Questão 2

Os componentes e as interfaces estão intimamente relacionadas. Enquanto eles são as peças principais a partir das quais os aplicativos são construídos, elas constituem as suas partes externamente visíveis e utilizadas para conectá-los. Em relação à variabilidade de interfaces, é correto afirmar que

A

uma interface só pode ser fornecida por um componente.

B

o uso de diferentes algoritmos ou protocolos não tem efeito sobre o projeto de uma interface.

C

o fornecimento de funções ou parâmetros que indiquem o nível de certos recursos pode ser necessário no projeto de uma interface para lidar com a variabilidade dos recursos.

D

não existe a possibilidade de a variabilidade estar relacionada à configuração do aplicativo; consequentemente, isso não influencia no projeto de uma interface.

E

processos de inicialização e de tratamento de erros nunca compartilham uma mesma interface proporcionada por diferentes componentes.

A alternativa C está correta.

A variabilidade pode estar representada em interfaces e componentes. No caso delas, podemos citar quatro tipos de variabilidade que surtem efeitos no projeto de uma interface: implementação de diferentes algoritmos, diferentes recursos fornecidos, diferentes configurações do aplicativo e diferentes componentes ofertando a mesma interface.

Artefatos de testes

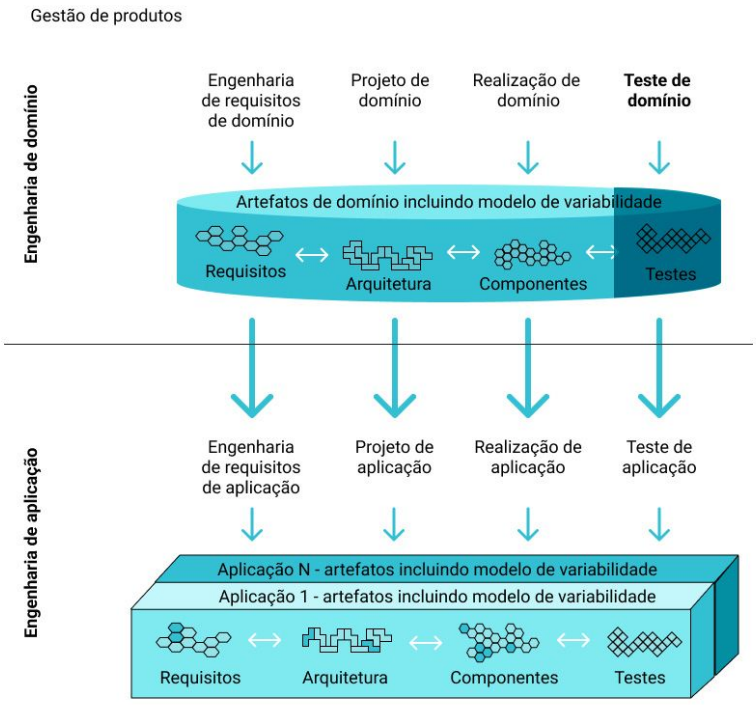
Importância do teste de software

A seguir, assista ao vídeo e confira os principais pontos sobre este tópico.

> Conteúdo interativo
>
> Acesse a versão digital para assistir ao vídeo.

O teste de software normalmente é a etapa derradeira do processo de construção de um aplicativo e tem o objetivo de verificar o seu nível de qualidade. Os erros passíveis de serem identificados pelos testes podem ser, por exemplo, erros de compatibilidade, de algoritmo e de requisitos não contemplados, assim como limitação de hardware.

Os artefatos de testes contêm informações relativas ao que testar, como fazê-lo e como documentar os resultados, as quais, por sua vez, também são consideradas artefatos do tipo. Como tanto a engenharia do domínio quanto a da aplicação realizam testes, os artefatos de testes são criados em ambas. Esses artefatos proporcionam testes repetíveis e rastreáveis.



Foco da modelagem de variabilidade nos artefatos de testes.

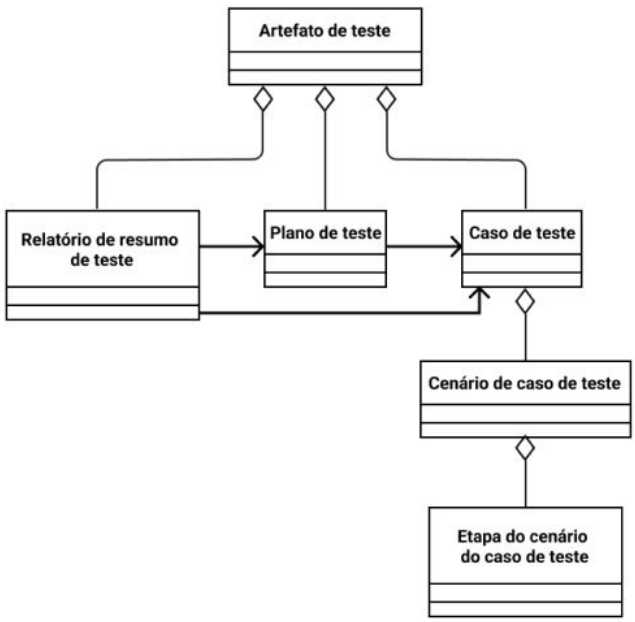
A imagem apresentada mostra o foco da documentação de variabilidade nos artefatos de testes, enquanto a que será apresenta mais adiante representa os artefatos de testes de software conforme o padrão IEEE 829 para documentação de teste de software.

Saiba mais

O padrão IEEE 829 especifica a forma de uso de um conjunto de documentos de teste de software.

A principal tarefa do teste na engenharia do domínio é o desenvolvimento de artefatos que possam ser reutilizados de maneira eficiente no teste de aplicações. Uma clara e inequívoca documentação da variabilidade nos artefatos de teste é fundamental para que essa tarefa seja bem-sucedida.

Faremos neste módulo uma breve descrição dos artefatos de testes mais importantes, mostrando ainda como o MOV pode ser empregado de forma integrada a esses artefatos.



Artefatos de testes de software segundo o IEEE.

Os artefatos de teste são produtos dos processos de teste, compostos, por sua vez, por planos, especificações e resultados. Apresentaremos a seguir um resumo sobre os tipos de artefatos de teste, que são as subclasses da imagem apresentada.

Tipos de artefatos de teste

Assista ao vídeo a seguir com os principais pontos sobre os tipos de artefatos de teste.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Relatório de resumo do teste

O relatório de resumo do teste proporciona uma visão geral dos resultados da execução de um teste específico. Esse relatório refere-se a:

Sistema testado

Plano de teste relacionado

Casos de teste executados

Ele documenta os resultados da execução de cada caso de teste e fornece o resumo geral de uma execução de teste, que pode incluir, por exemplo, uma classificação dos defeitos detectados.

Plano de teste

O plano de teste determina os casos de teste que deverão ser realizados, além de designar as prioridades para os casos e especificar quais recursos devem ser alocados e as ferramentas a serem utilizadas. Além de resumir o documento, explicando tanto seu propósito quanto o do sistema testado, sua introdução faz referência a outros documentos associados do projeto.

> Ela relaciona os itens de teste, descrevendo todos os elementos que devem ser testados, assim como as funcionalidades a serem testadas e aquelas que não devem sê-lo. Define, dessa forma, o escopo da testagem.

A estratégia do teste também está presente no plano de testes. Normalmente, há uma para cada grupo de funcionalidades contempladas. Questões, como, por exemplo, atividades, ferramentas usadas, critérios de sucesso ou falha, critérios de suspensão e requisitos de reinício, também são comuns no plano, além de outras informações.

Caso de teste

Os casos de teste definem as condições para que os testes sejam realizados de forma apropriada. Para isso, as seguintes informações são necessárias:

- Os dados de entrada;
- Os dados esperados na saída;
- Os objetivos daquele caso;
- O ambiente necessário;
- Um ou múltiplos cenários de testes;
- Informar como o caso de teste deve ser realizado.

Outra importante informação diz respeito aos critérios de falha, que definem se o artefato testado passou ou não no teste.

Há dois níveis de abstração para os casos de teste: lógico e detalhado. O primeiro cuida da descrição dos tipos de dados, das condições e das ações, sem descrever os dados a serem utilizados. O segundo nível proporciona os detalhes necessários para a realização dos casos de teste, eliminando qualquer ambiguidade de interpretação.

Cenário de caso de teste

Nesse artefato, são descritas as várias sequências específicas de ações que um caso de teste pode ter. Cada cenário é uma dessas sequências.

A execução de todos os cenários definidos para um caso de teste resulta na obtenção do objetivo daquele caso. Cada ação das sequências é um passo de cenário, conforme aponta a Artefatos de testes e software segundo o IEEE.

Passo do cenário de caso de teste

Cada passo de um cenário de um caso de teste contém instruções para a realização de uma ação da sequência de ações de cada cenário de caso de teste. Também pode conter, caso seja importante, os resultados esperados para aquele passo em particular.

Variabilidade nos artefatos de testes

Plano de teste

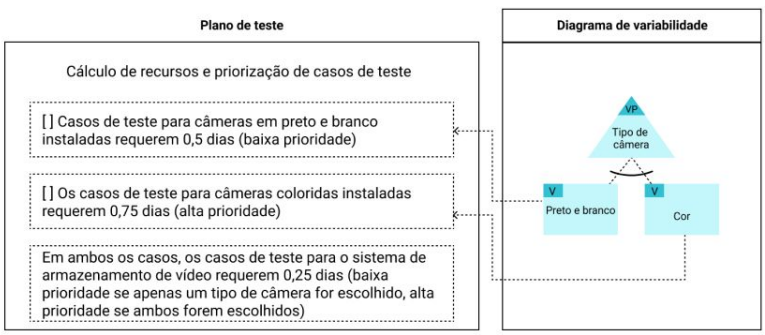
O plano de teste é utilizado na engenharia de domínio e na de aplicação. No caso da engenharia de domínio, esse plano deve determinar as atividades a serem realizadas de forma inequívoca, não contendo qualquer tipo de variabilidade.

No entanto, planos de testes genéricos para uso nos processos de testes da aplicação podem ser preparados. Nesses casos, eles sofrem o impacto da modelagem de variabilidade produzida pela ELPS.

> O plano de teste tem de estimar, pelo menos de forma aproximada, o tipo e a quantidade de recurso a ser utilizado nos casos de testes para cada variante. Os casos de testes comuns e variáveis também são definidos, e o plano precisa conter a informação sobre quais variantes devem ser ligadas aos planos variáveis.

A prioridade relativa dos casos de testes tem de ser explicitada conforme a variante a ser testada. Já as ferramentas a serem utilizadas, caso não deem suporte à variabilidade, devem ser complementadas com notas textuais ou outro meio de informação adequado. Como é especificada em linguagem natural, a maneira de modelar a variabilidade é a mesma utilizada (e vista anteriormente) nos requisitos textuais.

A imagem a seguir apresenta um exemplo de modelagem de variabilidade utilizando o MOV ligado a uma parte de um plano de teste genérico:



Exemplo de modelagem ortogonal de variabilidade em um plano de testes genérico.

Casos de teste

São os artefatos de testes mais importantes, sejam eles comuns ou variáveis. Os dados utilizados nos casos de teste devem ser adequados à variável considerada pelo MOV. Já o ambiente de software (configuração) e de hardware tem de ser apropriado aos itens de software testados. Por fim, os critérios de falha para cada item precisam ser definidos e ligados aos respectivos itens.

Sendo um artefato também escrito em linguagem natural, a documentação da variabilidade será semelhante à utilizada na modelagem dos requisitos textuais e do plano de teste.

Cenários de casos de teste

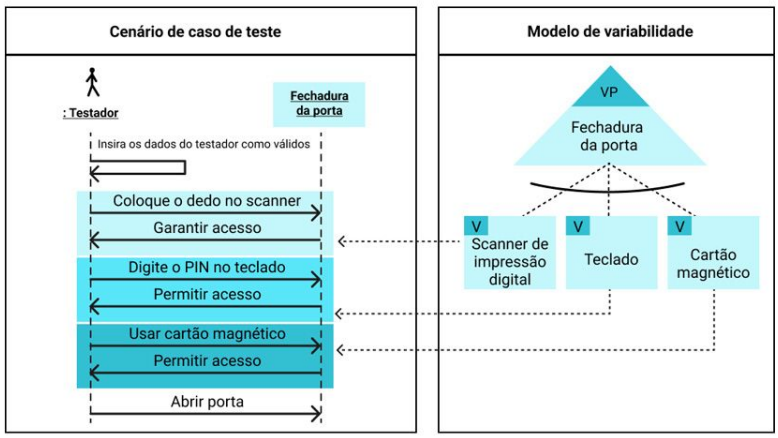
Contêm os passos que devem ser executados para a realização de um caso de teste, ordenando os passos de cenários de teste apropriadamente. A ELPS trabalha com três tipos de cenários:

Comum a todas as aplicações

Específico para uma variante

Adaptável a duas ou mais variantes

A imagem a seguir mostra o exemplo de um cenário de caso de teste representado como um digrama de sequência para o caso do travamento de porta do sistema de automação residencial. Ele é composto por um passo comum e por outros passos variáveis. Devido aos passos de inicialização e finalização comuns, tal cenário se encaixa na categoria “adaptável a duas ou mais variantes”.



Exemplo de modelagem ortogonal de variabilidade em um cenário de caso de testes.

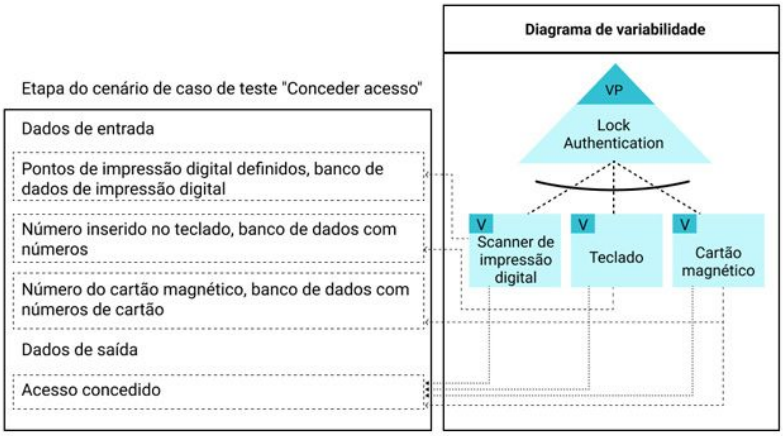
A especificação dos cenários de caso de teste é mais detalhada que a de cenários usada na engenharia de requisitos, em que pese o fato de ambas apresentarem as mesmas notações.

Passos de cenários de casos de teste são os passos individuais, isto é, as ações que compõem os cenários de tais casos. Esses passos contêm as especificações de entrada, as saídas esperadas e as informações de execução, e tudo isso pode variar conforme a variante sendo testada.

Exemplos de ações podem incluir “como apertar um botão específico”, “clicar com o botão direito do mouse” ou “navegar até uma linha específica em um arquivo de log”. Como é possível que essas ações dependam das variantes selecionadas, as informações de execução também variam. Observe as imagens a seguir:

Conteúdo interativo

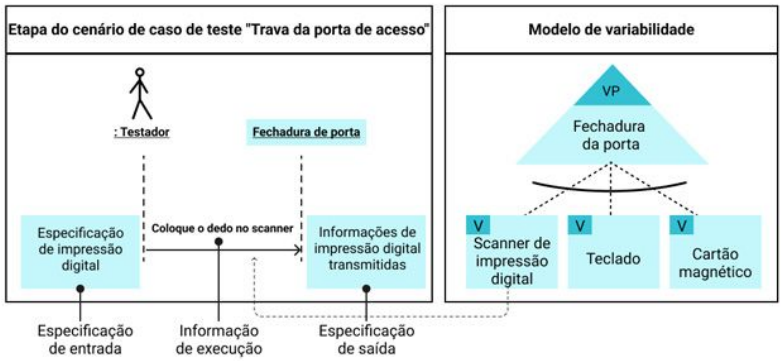
Acesse a versão digital para ver mais detalhes da imagem abaixo.



Exemplo de modelagem ortogonal de variabilidade em um passo de cenário de casos de teste com a utilização do MOV e representados em linguagem natural.

Conteúdo interativo

Acesse a versão digital para ver mais detalhes da imagem abaixo.



Exemplo de modelagem ortogonal de variabilidade em um passo de cenário de casos de teste com a utilização do MOV e representados em um diagrama de sequência.

Essas imagens apresentam exemplos de passos de cenário de casos de teste com a utilização do MOV. Sendo a anterior a representação em linguagem natural e a imagem a seguir a representação em diagrama de sequência.

Relatório de resumo do teste

O relatório de resumo de teste deve referenciar as variantes presentes no MOV, mas ele próprio não apresenta variabilidade. Desse modo, esse relatório pode ser documentado da mesma forma utilizada na engenharia de software tradicional. Ele precisa, porém, conter informações adicionais para referenciar as variantes.

A descrição do sistema testado pode incluir as variantes. Os casos de testes são descritos em relação àquelas testadas, enquanto as classificações dos defeitos distinguem os defeitos dos artefatos do domínio e da aplicação.

Variabilidade nos artefatos de testes

A seguir, assista ao vídeo e confira os principais pontos sobre este tópico.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Verificando o aprendizado

Questão 1

O teste de software é a etapa do processo de construção de um aplicativo cujo objetivo é verificar o nível de qualidade dos artefatos produzidos, e os artefatos de teste são produtos dos processos de teste. Em relação aos tipos de artefatos de teste, é correto afirmar que

A

o relatório de resumo do teste determina os casos de teste que deverão ser realizados, além de designar as prioridades para os casos e especificar quais recursos devem ser alocados e as ferramentas a serem utilizadas.

B

existem três níveis de abstração para os casos de teste: lógico, físico e detalhado.

C

em um cenário de caso de teste, é descrita a sequência específica de ações que o caso de teste tem de seguir para aquele cenário.

D

o tipo e a quantidade de recurso a ser utilizado nos casos de testes para cada variante devem ser informados no plano de teste.

E

o plano de teste proporciona uma visão geral dos resultados da execução de um teste específico.

A alternativa C está correta.

Um cenário de caso de teste contém os passos que devem ser executados para a realização de um caso de teste, ordenando os passos de cenários de teste apropriadamente em uma sequência. Um caso de teste pode ter vários cenários, sendo cada um deles representado por uma sequência de ações.

Questão 2

A variabilidade de uma LPS deve ser refletida nos artefatos de testes por meio do uso do MOV, adequando sua utilização conforme o artefato. Quanto à documentação de variabilidade nesses artefatos, pode-se afirmar que

A

é comum utilizar um diagrama de sequência para documentar a variabilidade em um relatório resumo de teste.

B

a configuração do ambiente do software é documentada nos casos de teste.

C

existem dois tipos possíveis de cenários de casos de testes: específico para uma variante e comum a todas as aplicações.

D

não é possível a utilização de um diagrama de sequência para a documentação em um caso de teste.

E

o plano de testes contém a informação referente aos critérios de falha de cada item testado.

A alternativa B está correta.

Os casos de testes são os artefatos de testes mais importantes. O ambiente no qual o caso é executado deve estar adequado ao item de software testado, levando em conta a variante considerada. O ambiente refere-se à configuração do software e ao hardware necessário.

Considerações finais

Neste estudo, apresentamos, de forma sucinta, a importância do assunto no desenvolvimento de artefatos de software reutilizáveis em uma linha de produto de software. Mostramos ainda como documentar e gerenciar a variabilidade, que é uma das principais atividades da ELPS.

A elaboração de artefatos de desenvolvimento que apresentam características de variabilidade e o reaproveitamento em aplicações diversas da mesma linha de produtos se fundamentam na correta documentação da variabilidade. Com isso, frisamos que a ELPS enxerga a variabilidade como uma propriedade essencial dos artefatos de domínio, sendo a modelagem de variabilidade usada para capturar a variabilidade dos requisitos de domínio, arquitetura, componentes e testes.

Você agora reúne as condições para reconhecer os princípios da variabilidade da LPS e documentar a variabilidade de uma LPS em artefatos de requisitos, de projeto, de realização e de testes.

Podcast

Para encerrar, ouça a entrevista com os principais pontos deste estudo.

Conteúdo interativo

Acesse a versão digital para ouvir o áudio.

Explore +

O assunto **variabilidade em engenharia de linha de produto de software** é vastamente documentado na Web. É muito interessante que você pesquise sobre o tema, buscando complementar e sedimentar o aprendizado.

Este artigo tem como objetivo discutir a variabilidade dinâmica da modelagem de software:

COUTINHO, E. F.; BEZERRA, C. I. M. **Um estudo sobre a variabilidade de Aspectos Dinâmicos no ecossistema de software educacional Solar**. Anais do Workshop em Modelagem e Simulação de Sistemas Intensivos em Software (MSSIS). Publicado em: 13 set. 2019.

O artigo a seguir trata da arquitetura de referência (AR) como um conceito cujo objetivo é concentrar as funcionalidades de um domínio de aplicação específico, fornecendo uma estrutura arquitetural genérica para o desenvolvimento de novos sistemas ou a evolução daqueles existentes. Atualmente, diversas ARs vêm sendo propostas para diferentes contextos. A VMTools-RA, uma AR para ferramentas de variabilidade de software, foi projetada nesse sentido.

SILVA, L. F. da; OLIVEIRA JR., E. **Avaliação empírica da VMTools-RA**: uma arquitetura de referência para ferramentas de variabilidade de software. Anais do Workshop em Modelagem e Simulação de Sistemas Intensivos em Software (MSSIS). Publicado em: 25 set. 2019.

Referências

CLEMENTS, P.; NORTHROP, L. **Software product lines**: practices and patterns. 3. ed. Addison-Wesley Professional, 2001.

POHL, K.; BOCKLE, G.; LINDEN, F. **Software product line engineering**: foundations, principles, and techniques. Essen: Springer, 2005.

POHL, K.; BOCKLE, G.; LINDEN, F. **Software product line engineering**: foundations, principles, and techniques. Essen: Springer, 2011.

SILVA, F. A. P. da. *et al*. **Linhas de produtos de software**: uma tendência da indústria. Sociedade Brasileira de Computação, 2011.

SOFTWARE ENGINEERING INSTITUTE. **A framework for software product line practice** – version 5.0. 2016.