




Variabilidad en procesos software


Universidad de Castilla-La Mancha
Escuela Superior de Informática
Departamento de Tecnologías y Sistemas de Información



Introducción

- **Introducción**
- Líneas de procesos software
 - Justificación y necesidad
- Modelado de Líneas de Proceso
 - SPEM
 - Descripción general
- Variabilidad en SPEM
- Variabilidad en líneas de proceso software


2



Introducción

- Desarrollo de software orientado a procesos
- Debido a la demanda de software de calidad
 - Ingeniería de procesos
 - Aplicación de los principales paradigmas de programación a los procesos software
 - Especialmente el Desarrollo dirigido por Modelos
- Esfuerzos para mejorar la capacidad del software
 - Desarrollo de modelos de procesos
 - CMM, CMMI, ISO 12207
 - Desarrollo de Metodologías
 - Métrica, RUP
 - Desarrollo de estándares para modelar procesos
 - SPEM


3



Introducción (II)

- Esfuerzos orientados a la genericidad
- Necesidad de adaptar los modelos de procesos y metodologías antes de aplicarlos
 - Gran cantidad de organizaciones desarrolladoras de software
 - “al igual que no hay dos proyectos iguales, tampoco hay dos procesos iguales en el mundo” [Humphrey, 1989]
- Durante la adaptación de los procesos se crean familias de procesos
- Líneas de Procesos software
 - Buen uso de las similitudes entre procesos
 - Explotar las diferencias entre ellos


4



Introducción (III)

- Modelado de Líneas de procesos software
 - Modelar estructura común
 - Modelar variaciones
- SPEM (Software Process Engineering Metamodel)
 - Orientado al modelado de procesos
 - No de Líneas de proceso
 - No soporta la variabilidad específica de las Líneas de Procesos Software
- Necesidad de definir variabilidad específica sobre los procesos de SPEM

5



Indice

- Introducción
- **Líneas de procesos software**
 - **Justificación y necesidad**
- Modelado de Líneas de Proceso
 - SPEM
 - Descripción general
- Variabilidad en SPEM
- Variabilidad en líneas de proceso software

6

Líneas de Proceso Software

- **Ingeniería de Métodos**
 - Aplicación de la ingeniería del software a procesos
 - Evolución, mejora de las metodologías de desarrollo software
- **Necesita evolucionar**
 - Al igual que las técnicas de desarrollo de productos
 - Desarrollo de Software basado en Componentes
 - Permitir la adaptación de procesos, no métodos

Variaciones debidas a la Ejecución localizadas en el Proceso

```

    graph LR
      Metodo[Método] --> Proceso((Proceso))
      Proceso --> Adaptaciones{Adaptaciones Ejecución}
      Adaptaciones --> ProcesoA([Proceso A])
      Adaptaciones --> ProcesoB([Proceso B])
      Adaptaciones --> ProcesoC([Proceso C])
  
```

7

Líneas de Proceso Software

Líneas de Producto Software

Artefactos para extraer beneficios a un conjunto de productos software

- A través de sus diferencias y partes en común
- Reducción de costos

Dos etapas

- Ingeniería
- Dominio

Puntos de variación

Variantes

Core product

The graph plots 'Esfuerzo Total' on the y-axis against '# Productos' on the x-axis. The y-axis also includes markers for 'Coste Unitario (LPS)', 'Inversión Inicial', and 'Coste Unitario (convencional)'. A red dashed line represents 'Desarrollo Convencional', showing a linear increase in total effort. A blue solid line represents 'Desarrollo de Línea de Producto', which starts with a higher initial investment but then levels off, crossing below the conventional line at the third product. A yellow circle highlights the intersection point labeled 'Retorno'. The area between the lines from product 4 onwards is shaded and labeled 'Ahorro'.

8

Líneas de Proceso Software
Líneas de Producto Software

Ejemplo:

- Calculadora
- Coreprocess: estructura general (botones, números, pantalla)
- Punto de variación: funciones a implementar
- Variantes: diferentes funciones

Línea de Procesos Software

- Línea de Procesos Software
 - Aplicación del enfoque de Líneas de Productos Software en los procesos software
 - Gestión de las similitudes y diferencias entre un conjunto de procesos software
 - Reuso y automatización
 - Propagación de las mejores prácticas
 - Procesos adaptados a cada proyecto
 - Facilitando su implantación en las organizaciones
- Puede adoptar los mecanismos de variación de las Líneas de Productos Software
 - Parte común a todos los procesos
 - Parte variable
 - A través de variaciones **puntuales**

11



Línea de Procesos Software

- Modelado de variabilidad a través de
 - Puntos de variación
 - Puntos del modelo de proceso donde ocurre la variación
 - Variantes
 - Elementos que representan la variabilidad, y ocupan los puntos de variación
 - Relaciones
 - Para vincular puntos de variación y variantes
 - Dependencias y restricciones
 - Para asegurar la consistencia del modelo de procesos a crear
- Estos elementos clave deben incluirse en el metamodelo que dé soporte a las Línea de Procesos Software

12



Índice

- Introducción
- Líneas de procesos software
 - Justificación y necesidad
- **Modelado de Líneas de Proceso**
 - **SPEM**
 - Descripción general
- Variabilidad en SPEM
- Variabilidad en líneas de proceso software

13

SPEM
Características básicas

- SPEM (Software Process Engineering Metamodel)
- Metamodelo para el modelado
 - De procesos
 - De métodos
 - De los que se derivan los procesos
- Metamodelo MOF-compliant
 - Está definido como un metamodelo del nivel M2 de MOF.
 - Reutiliza algunas clases de UML 2.
- Perfil de UML 2.
 - La definición sólo abarca la presentación, ya que las definiciones semánticas y restricciones están en el metamodelo.

14

SPEM
Características básicas

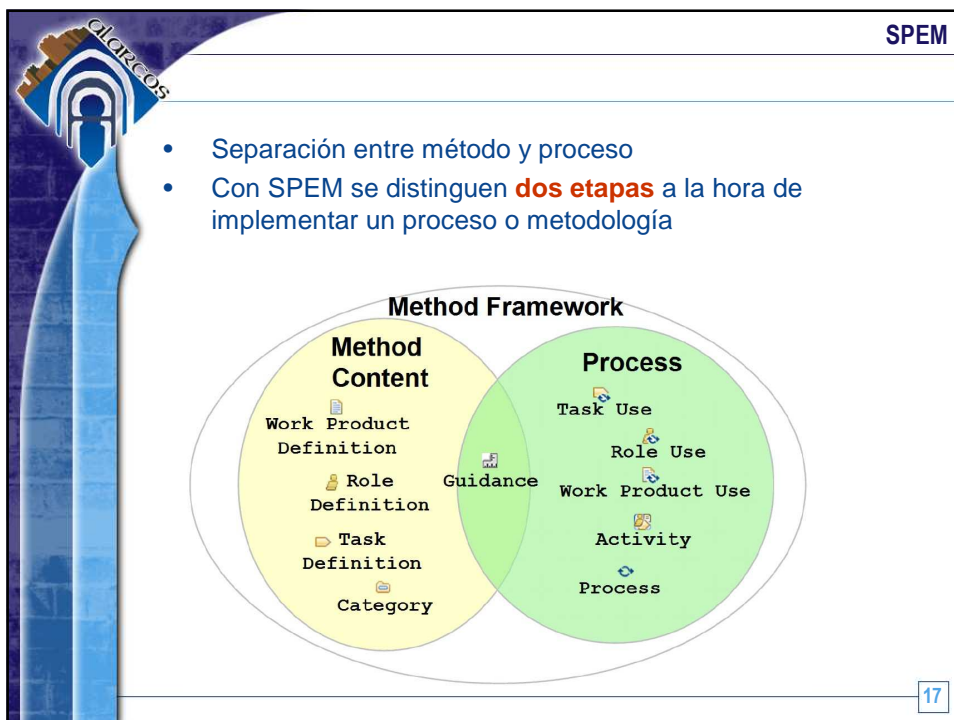
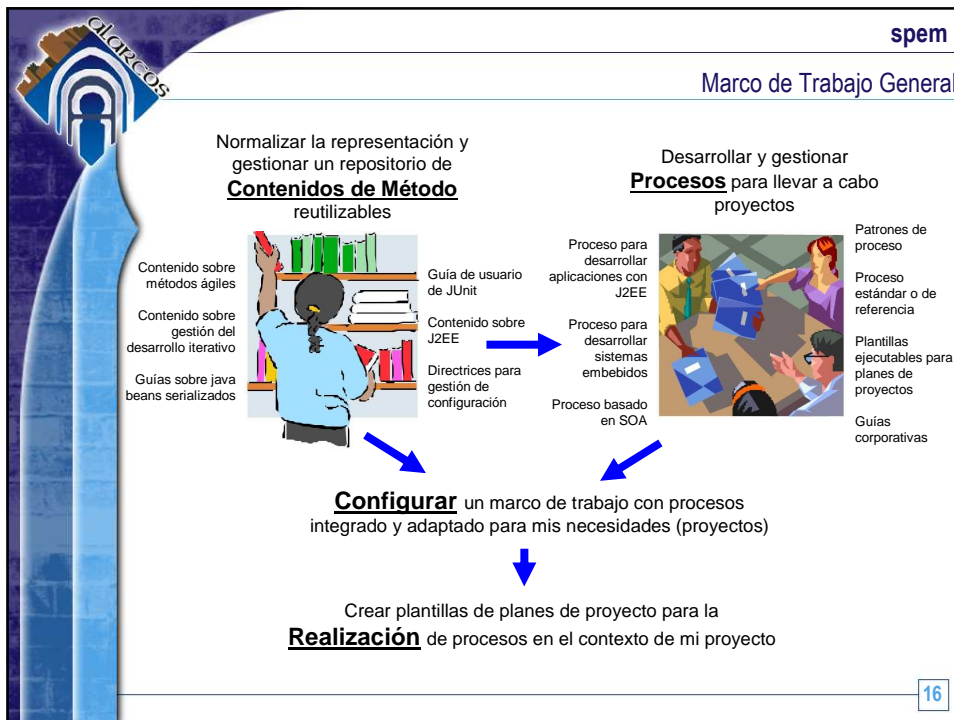
Idea básica de proceso:

```

classDiagram
    class Rol
    class Producto_de_Trabajo["Producto de Trabajo"]
    class Tarea

    Rol "1" -- "0..*" Producto_de_Trabajo : es responsable de
    Rol "1" -- "0..*" Tarea : realiza
    Producto_de_Trabajo "0..*" -- "0..*" Tarea : Usa
    Producto_de_Trabajo "0..*" -- "0..*" Tarea : Produce
    Producto_de_Trabajo "0..*" -- "0..*" Tarea : +entrada
    Producto_de_Trabajo "0..*" -- "0..*" Tarea : +salida
  
```

15



SPEM
Paquetes

- SPEM se estructura en 7 paquetes
 - Estructurados jerárquicamente

```

    graph TD
      MethodPlugin -- «merge» --> ProcessWithMethods
      MethodPlugin -- «merge» --> MethodContent
      ProcessWithMethods -- «merge» --> ProcessStructure
      ProcessWithMethods -- «merge» --> MethodContent
      ProcessBehavior -- «merge» --> ProcessStructure
      ProcessBehavior -- «merge» --> MethodContent
      ProcessStructure -- «merge» --> Core
      MethodContent -- «merge» --> ManagedContent
      ManagedContent -- «merge» --> Core
      
```

18

SPEM
Core

- Paquete Core
 - Contiene las abstracciones necesarias para estructurar el resto de componentes
 - *ExtensibleElement*
 - *Kind*
 - *WorkDefinition*
 - Asociado con Restricciones
 - *WorkDefinitionPerformerMap*
 - *WorkDefinitionParameter*

19

SPEM
Process Structure

- **Paquete Process Structure**
 - Incluye los elementos necesarios para definir los modelos de procesos
 - *ProcessElement*
 - *BreakdownElement*
 - *WorkBreakdownElement*
 - Activity
 - WorkproductUse
 - RoleUse
 - Milestone
 - WorkSequence

20

SPEM
Process Structure

Elemento de Desglose – Breakdown Element

- Es una generalización abstracta para cualquier tipo de elemento que aparece en un proceso y es parte de una estructura de desglose.
- Tienen tres propiedades importantes:
 - Admite **Varias Apariciones** (Has Multiple Occurrences)
 - Es **Opcional** (Is Optional)

```

classDiagram
    class SPEM_Core_WorkDefinition["SPEM:Core::WorkDefinition"]
    class SPEM_Core_WorkDefinitionParameter["SPEM:Core::WorkDefinitionParameter"]
    class SPEM_Core_WorkDefinitionPerformerMap["SPEM:Core::WorkDefinitionPerformerMap"]
    class SPEM_Core_ExtendableElement["SPEM:Core::ExtendableElement"]
    class ProcessElement
    class BreakdownElement
    class ProcessParameter
    class ProcessPerformerMap
    class WorkSequence
    class WorkBreakdownElement
    class RoleUse
    class WorkProductUse
    class Activity
    class Milestone
    class WorkProductUseRelationship
    class ProcessResponsibilityAssignmentMap

    SPEM_Core_WorkDefinition <|-- ProcessElement
    SPEM_Core_WorkDefinitionParameter <|-- ProcessElement
    SPEM_Core_WorkDefinitionPerformerMap <|-- ProcessElement
    SPEM_Core_ExtendableElement <|-- ProcessElement
    ProcessElement <|-- BreakdownElement
    BreakdownElement <|-- ProcessParameter
    BreakdownElement <|-- ProcessPerformerMap
    BreakdownElement <|-- WorkSequence
    BreakdownElement <|-- WorkBreakdownElement
    BreakdownElement <|-- RoleUse
    BreakdownElement <|-- WorkProductUse
    Activity <|-- ProcessParameter
    Activity <|-- ProcessPerformerMap
    Activity <|-- WorkSequence
    Activity <|-- WorkBreakdownElement
    Milestone <|-- WorkBreakdownElement
    WorkBreakdownElement --> RoleUse
    WorkBreakdownElement --> WorkProductUse
    WorkBreakdownElement --> WorkProductUseRelationship
    WorkBreakdownElement --> ProcessResponsibilityAssignmentMap
  
```

21



SPEM
Process Structure

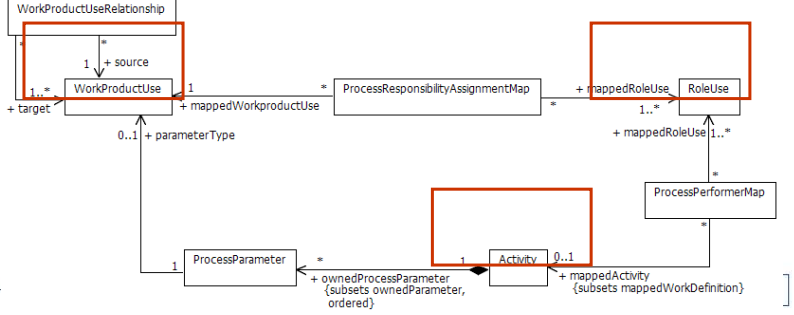
Elemento de Desglose de Trabajo - Work Breakdown Element

- Es un Elemento de Desglose que representa Trabajo.
- Existen dos tipos:
 - Actividad
 - Hito
- Propiedades:
 - Se Puede Repetir
 - Continuo (Is Ongoing):
 - Condicionado por Sucesos (Is Event Driven):

22

SPEM
Process Structure

- **WorkProductUse:** 
 - Representa un producto de trabajo de entrada o salida, relacionado con una actividad
- **RoleUse:** 
 - es el ejecutor de una actividad o un participante en la misma.
- Existen relaciones específicas entre ellos y actividad



```

classDiagram
    class WorkProductUseRelationship {
        + source WorkProductUse
        + target WorkProductUse
    }
    class WorkProductUse {
        + parameterType
    }
    class RoleUse {
        + mappedRoleUse
    }
    class Activity {
        + ownedProcessParameter
        + mappedActivity
    }
    class ProcessParameter {
    }
    class ProcessResponsibilityAssignmentMap {
    }
    class ProcessPerformerMap {
    }

    WorkProductUseRelationship "1" -- "*" WorkProductUse : source
    WorkProductUseRelationship "1" -- "*" WorkProductUse : target
    WorkProductUse "1" -- "*" ProcessResponsibilityAssignmentMap : mappedWorkproductUse
    RoleUse "1..*" -- "*" ProcessResponsibilityAssignmentMap : mappedRoleUse
    Activity "1" -- "*" ProcessParameter : ownedProcessParameter
    Activity "0..1" -- "*" ProcessPerformerMap : mappedActivity
  
```

SPEM
Process Structure

- El **Flujo** entre los Elementos de Desglose de Trabajo se representa por medio de **Secuencias de Trabajo**,
- Cada Secuencia de Trabajo, entre un predecesor P y un sucesor S, es de una de las siguientes clases:
 - Acabar para Empezar: S no puede empezar hasta que no concluye P.
 - Acabar para Acabar: S no puedo acabar mientras no esté acabado P.
 - Empezar para Empezar: S no puede comenzar hasta que no lo ha hecho P.
 - Empezar para Acabar: S no puede concluir hasta que no se inicia P.
 - Se emplea en Just-in-Time.


24

SPEM
Process Structure

- Ejemplo de una actividad con sus asociaciones

25

SPEM
Process Structure




Hito - Milestone

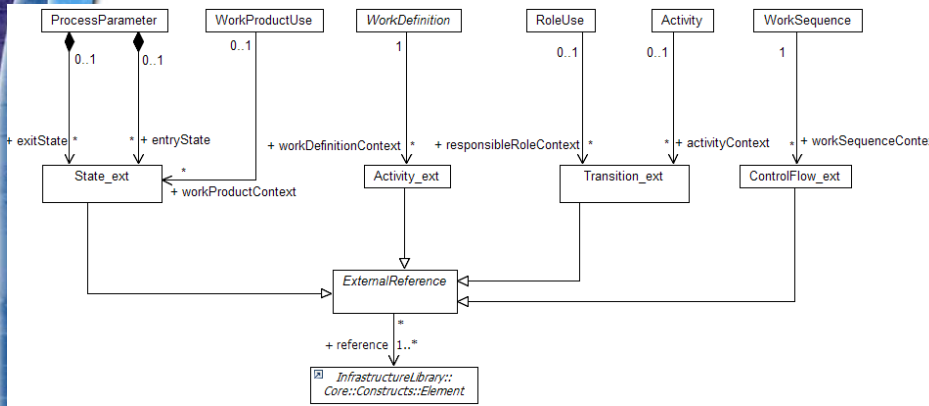
- Representa un evento significativo para el desarrollo de un proyecto:
 - Decisión importante
 - Conclusión de un entregable
 - Conclusión de una fase, ...
- Es un Elemento de Desglose de Trabajo, por tanto
 - Aparece en la estructura de desglose de trabajo, y
 - Puede tener relaciones de precedencia.

26

SPEM
ProcessBehavior



- ProcessBehaviour
 - Contiene los elementos para modelar el comportamiento de los modelos de procesos a través de referencias externas

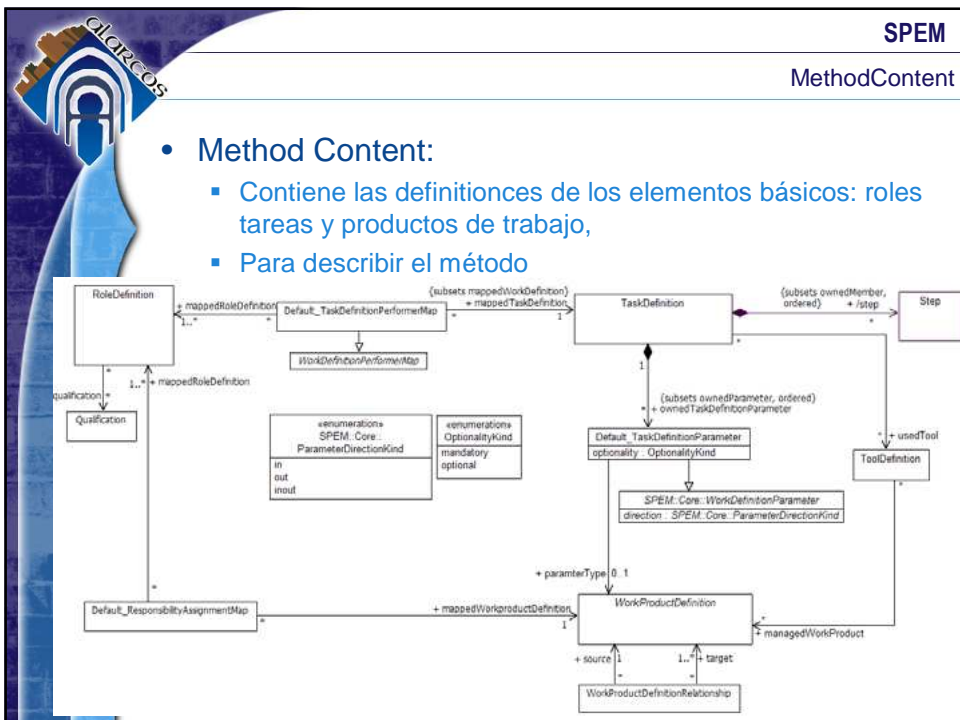
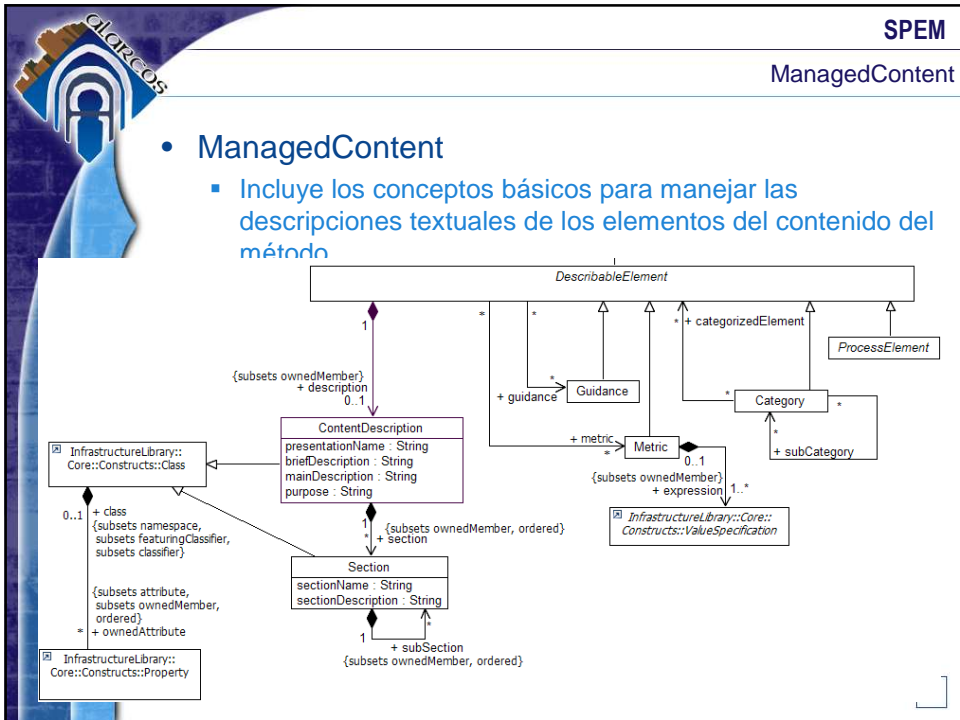


```

classDiagram
    class ProcessParameter
    class WorkProductUse
    class WorkDefinition
    class RoleUse
    class Activity
    class WorkSequence
    class State_ext
    class Activity_ext
    class Transition_ext
    class ControlFlow_ext
    class ExternalReference
    class InfrastructureLibrary["InfrastructureLibrary::Core::Constructs::Element"]

    ProcessParameter "0..1" --> "*" State_ext : + exitState
    State_ext "0..1" --> "*" ProcessParameter : + entryState
    WorkProductUse "0..1" --> "*" State_ext : + workProductContext
    WorkDefinition "1" --> "*" Activity_ext : + workDefinitionContext
    RoleUse "0..1" --> "*" Activity_ext : + responsibleRoleContext
    Activity "0..1" --> "*" Transition_ext : + activityContext
    WorkSequence "1" --> "*" ControlFlow_ext : + workSequenceContext
    State_ext --> "*" ExternalReference
    Activity_ext --> "*" ExternalReference
    Transition_ext --> "*" ExternalReference
    ControlFlow_ext --> "*" ExternalReference
    ExternalReference "*" --> "1..*" InfrastructureLibrary : + reference
  
```

27




SPEM
MethodContent



- **RoleDefinition** 
 - Es el conjunto de perfiles, competencias y responsabilidades de un individuo o conjunto de ellos
- **TaskDefinition** 
 - Identifica el trabajo que se ejecuta por los RoleDefinition
 - Puede dividirse en pasos (steps) 
- **ToolDefinition** 
 - Define las características de una herramienta que debe utilizarse para llevar a cabo la tarea
- **WorkProductDefinition** 
 - Es el producto usado o producido por las TaskDefinition

30

SPEM
MethodContent



- **Apariencia de un método en SPEM**

```

graph TD
    Task[Prioritize Use Cases]
    SA[Software Architect]
    SA ---|«performs, primary»| Task
    SA ---|«performs, additional»| Task
    SA ---|«performs, additional»| Task
    SA ---|«optional, input»| Task
    Task ---|«mandatory, input»| UCM[«work product definition, artifact»  
Use Case Model]
    Task ---|«mandatory, output»| RA[Requirements Attributes]
    Task ---|«optional, output»| SAD[«work product definition, deliverable»  
Software Architecture Document]
  
```



31

SPEM
ProcessWithMethods


- **ProcessWithMethods**
 - Permite la definición de estructuras de desglose (procesos) mapeándolos con elementos del contenido del

32

SPEM
ProcessWithMethods

- **Redefine algunos de los elementos de ProcessStructure**
 - BreakdownElement
 - Activity
 - RoleUse
 - WorkProductUse
- **TaskUse** 
 - Uso de una TaskDefinition en una actividad
- **ProcessPackage** 
 - Paquete de elementos de proceso


33



SPEM
MethodPlugin

- Method Plugin
 - Permite el diseño y el manejo de métodos y procesos reusables y configurable, a través de librerías
- Define
 - MethodPlugin
 - Contiene paquetes de métodos y procesos
 - MethodConfiguration
 - Colección de method plugins
 - MethodLibrary
 - Contenedor de MethodPlugins y MethodConfigurations
 - ProcessComponent
 - Proceso encapsulado del que sólo se conocen las entradas y salidas
- Elementos que permiten el reuso a través de la encapsulación

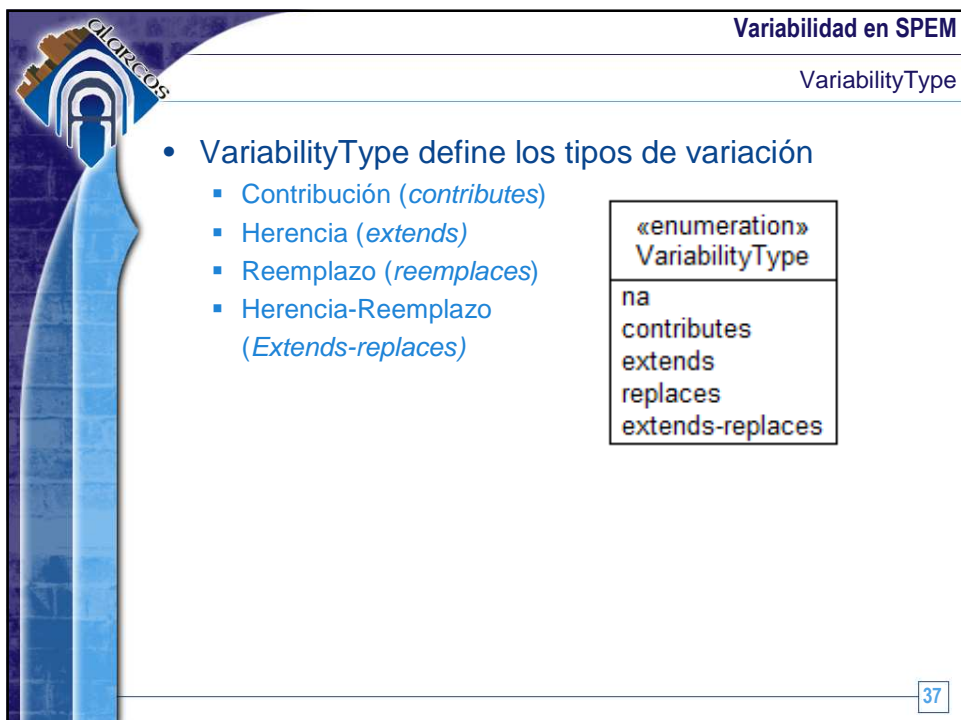
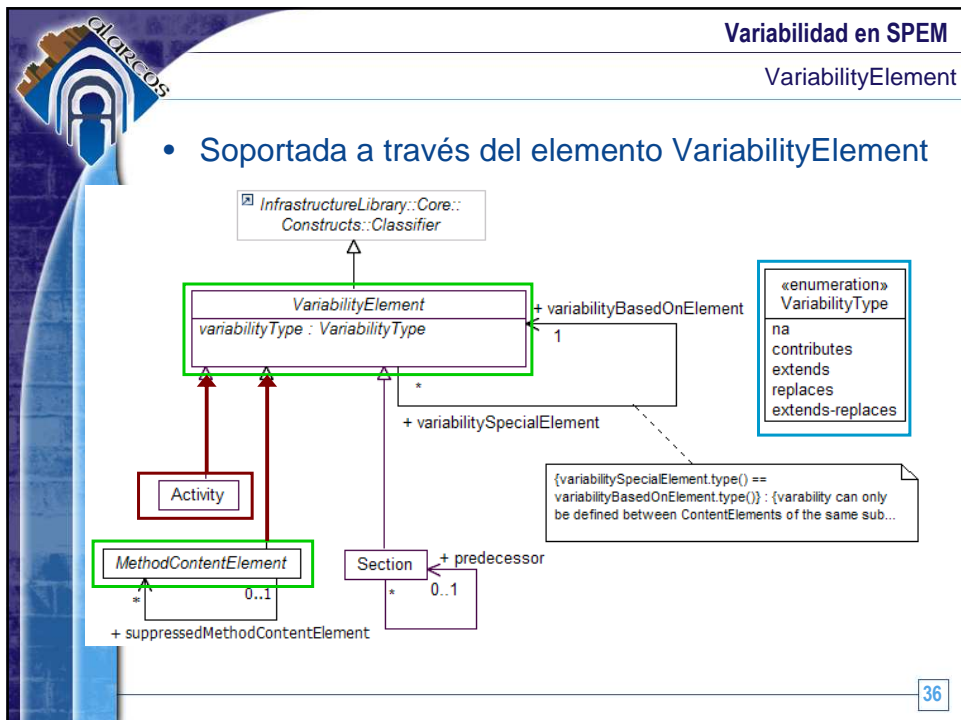
34



Introducción

- Introducción
- Líneas de procesos software
 - Justificación y necesidad
- Modelado de Líneas de Proceso
 - SPEM
 - Descripción general
- **Variabilidad en SPEM**
- Variabilidad en líneas de proceso software

35



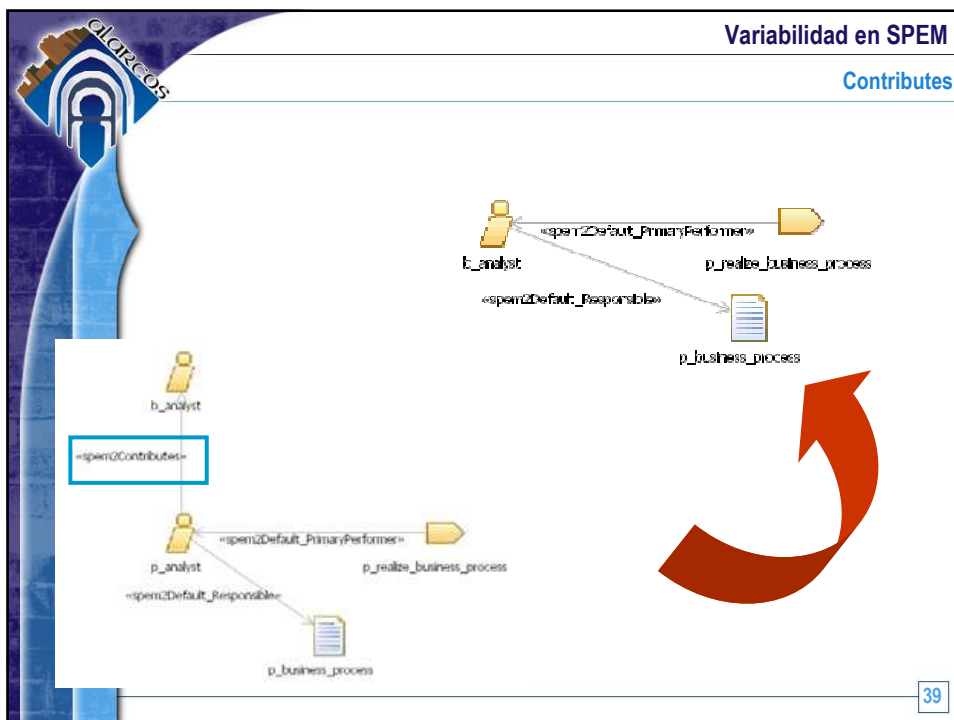
Variabilidad en SPEM
Contributes

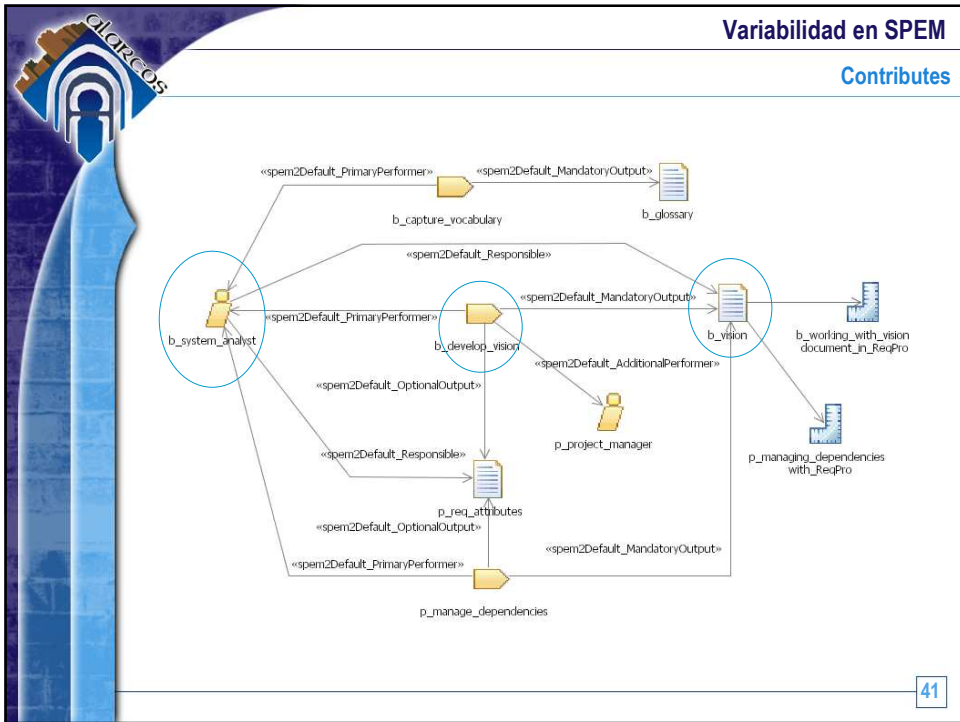
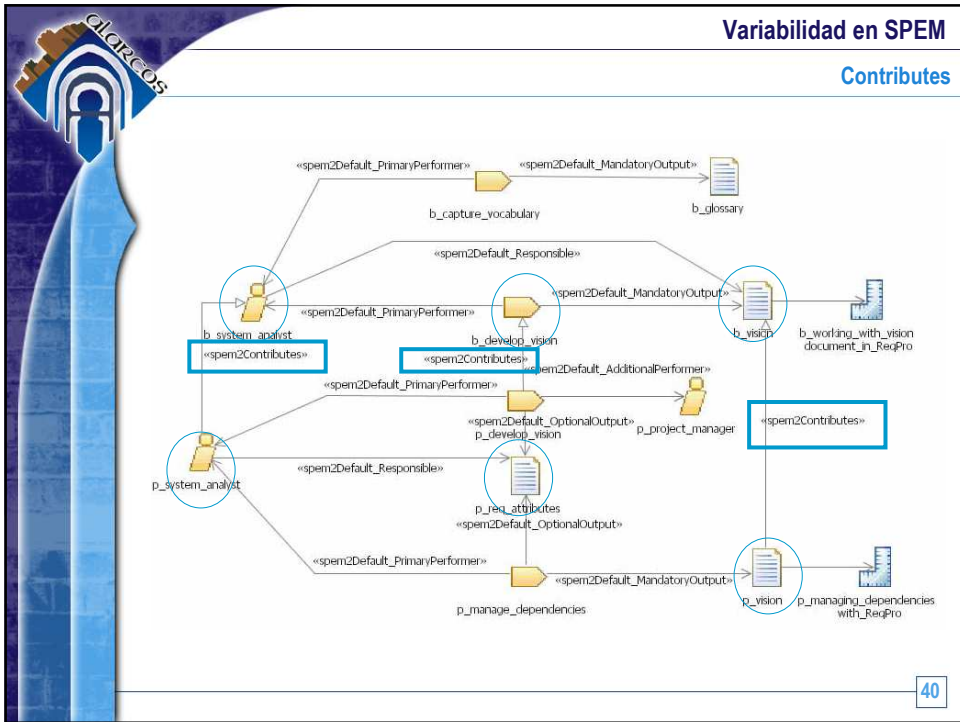
- Contributes: Añade las propiedades de un elemento E a otro EB, sin alterarlo. E no se muestra en las vistas generadas

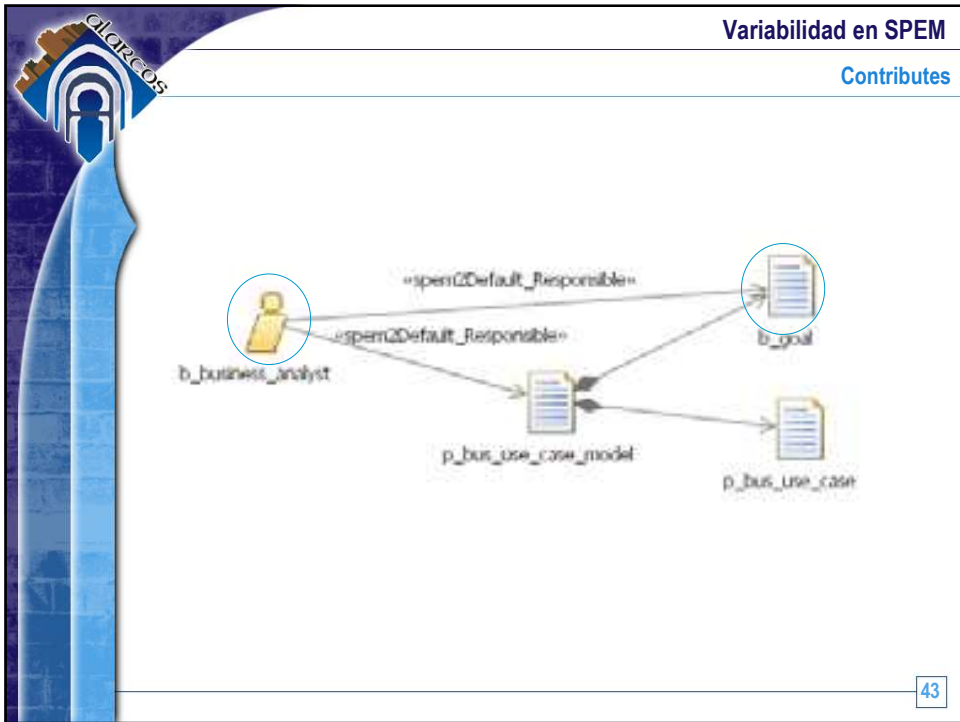
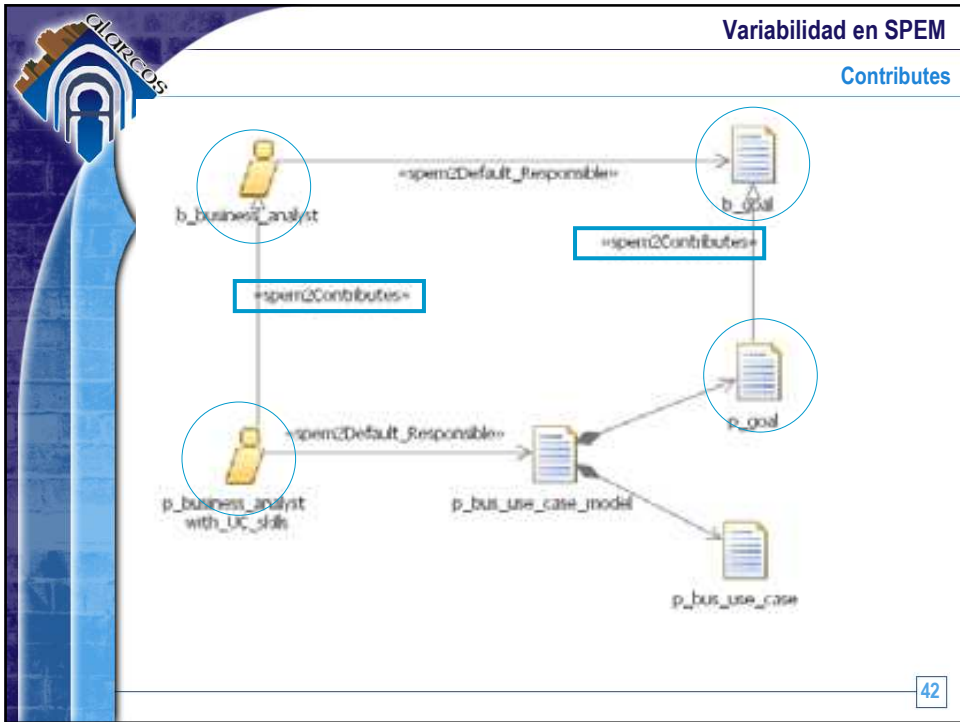
Reglas de contribución:

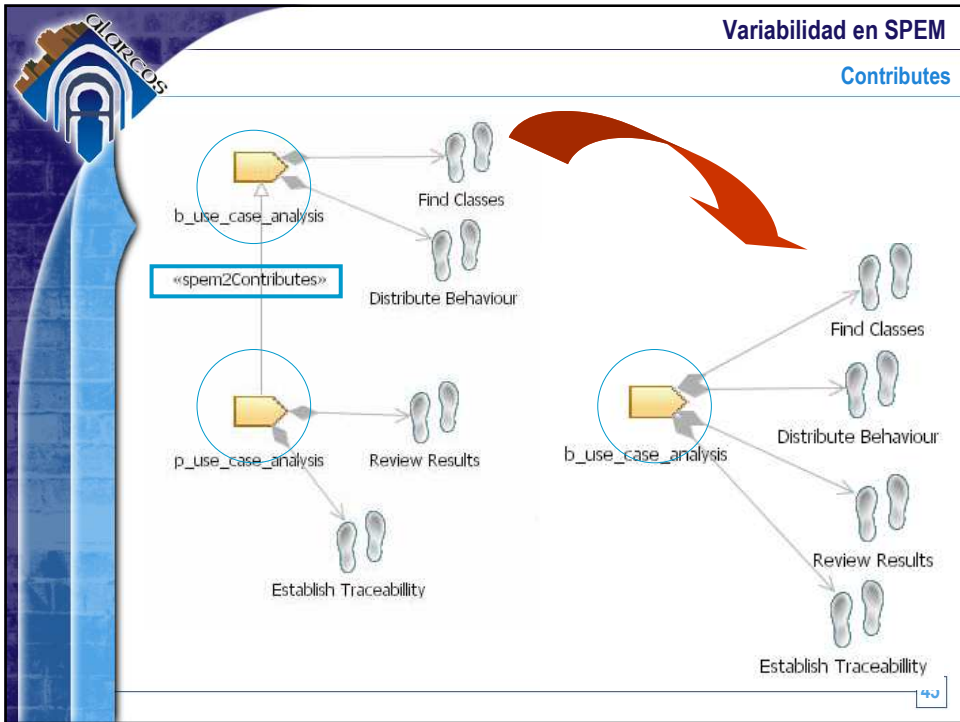
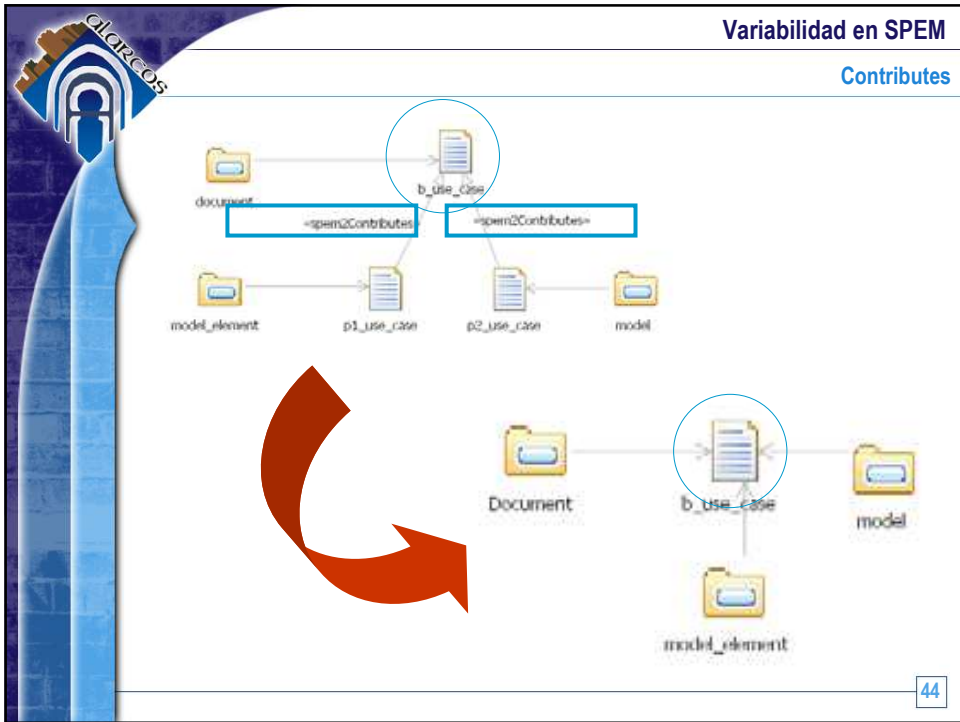
- Los atributos (campos de texto) del elemento que contribuye se concatenan al final de los respectivos campos en el elemento base.
- Las asociaciones de entrada y de salida "a muchos" del elemento que contribuye se añaden al elemento base.
- En las asociaciones de entrada y de salida "a 1" (como la relación "realizador principal" entre una tarea y un rol) definidas, la relación del elemento que contribuye se ignora si la asociación existe en el elemento base.
- Un elemento base puede recibir varias contribuciones.
- La contribución es transitiva.

38









Variabilidad en SPEM

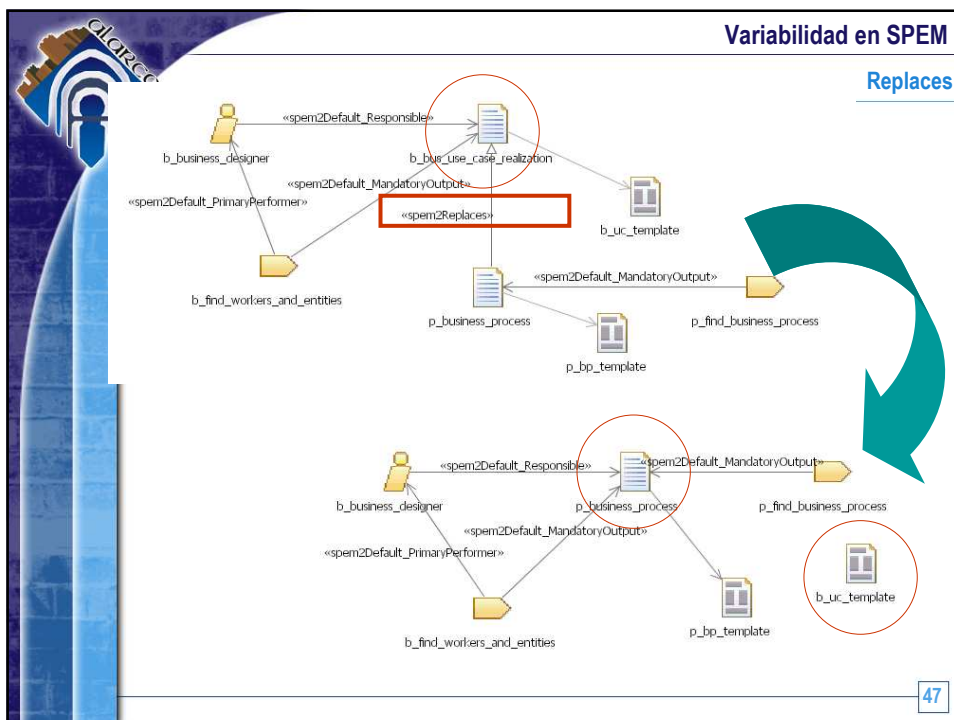
Replaces

- Replaces: Permite el reemplazo de un elemento EB por otro E, sustituyendo sus propiedades. El elemento EB no aparecerá en las vistas

Reglas

- Los valores de los atributos de EB se reemplazan por los respectivos valores de los atributos no vacíos de E, incluyendo los identificadores.
- Las instancias de asociación de salida de EB se reemplazan por las de E.
- Las instancias de asociación de entrada con cardinalidad “muchos” de EB se amplían con las instancias de dichas asociaciones existentes para E.
- Las instancias de asociación de entrada con cardinalidad “uno” de EB se dejan intactas. Las instancias incluidas en E que no están en EB se añaden a EB.
- Un elemento base sólo puede ser reemplazado por otro único elemento en una misma configuración. Si se define más de un reemplazo para un mismo elemento base, entonces no se realizará ningún reemplazo.
- El reemplazo es transitivo.

46



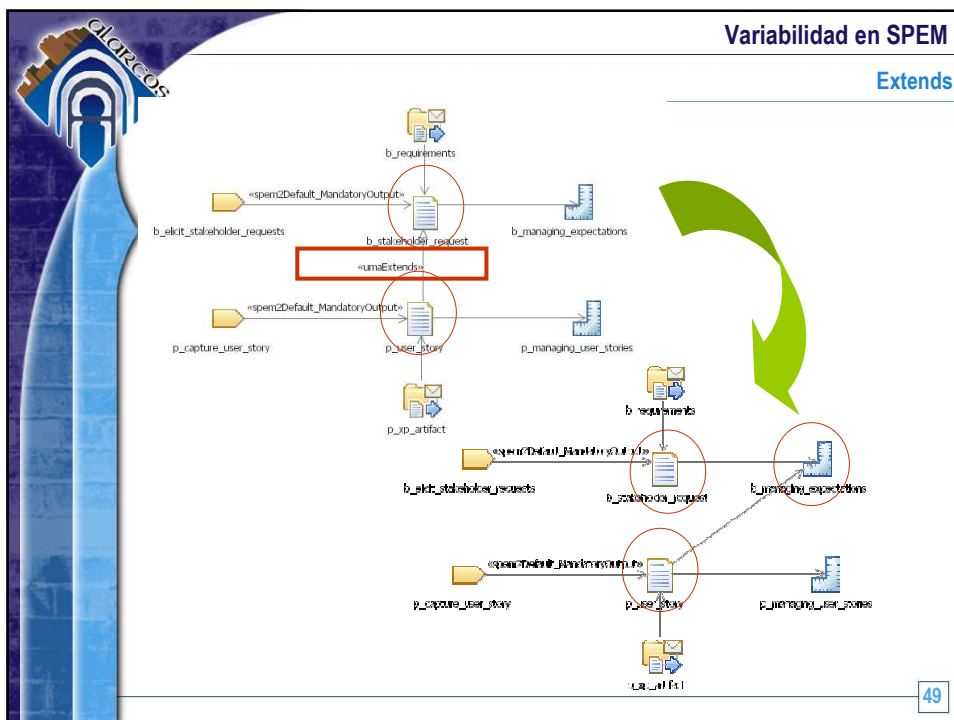
Variabilidad en SPEM
Extends

- Extends: Mecanismo de herencia donde el elemento E hereda todas las propiedades del elemento EB, sin alterar a EB.

Reglas

- Los valores de los atributos de EB son heredados por E en caso de que los equivalentes suyos estén vacíos. Si se definen campos en E, los respectivos campos de EB se ignoran.
- Las instancias de asociación de salida con cardinalidad “muchos” de EB se añaden a E.
- En las instancias de asociación de salida con cardinalidad “1” se ignora la instancia de EB si E ha definido la suya propia. En caso contrario, E hereda la de EB.
- Las asociaciones de entrada de EB se ignoran (no se añaden a E).
- Las asociaciones de ampliación son transitivas.

48



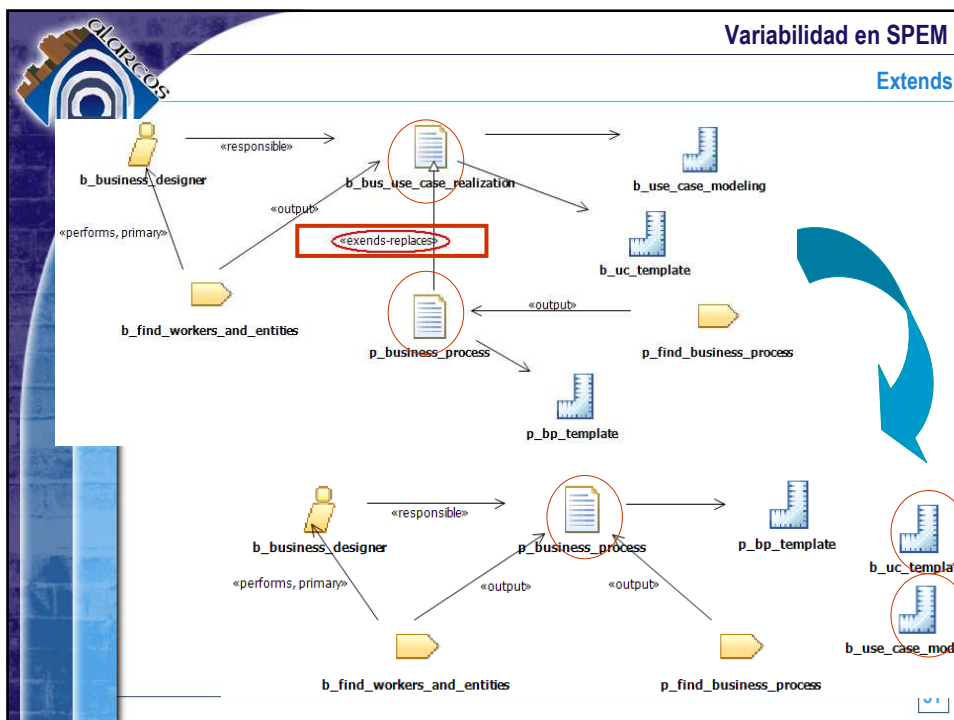
Variabilidad en SPEM
Extends-Replaces


- Extends-Replaces: Este tipo de variabilidad combina los efectos de *extends* y *replaces*, es decir amplía el elemento E con todas las propiedades de EB que no son definidas en E. EB no se considera en las vistas que se generan.

Reglas

- Si el elemento E que amplía y sustituye define sus propias asociaciones de salida, reemplazarán a las del elemento base EB. Las instancias de asociaciones de salida no definidas en E se heredan de EB.
- Las instancias de asociaciones de entrada de EB se añaden a E.
- Los valores de los atributos no vacíos de E reemplazan a los valores de los mismos atributos de EB. del elemento base. Si E no define valor para un atributo, el valor se hereda de EB.
- La relación amplía y reemplaza es transitiva.
- Un elemento base sólo puede ser reemplazado o ampliado y reemplazado por otro único elemento en una misma configuración. En caso contrario no se realizará ningún reemplazo.

50






Variabilidad en SPEM

Limitaciones a la variabilidad en SPEM

- Definida para variar métodos, no procesos
 - Todos los elementos usados para definir procesos pueden variar
 - Sólo las actividades pueden variar
 - Aun así, se puede variar el método del que se genera el proceso y a través de esto, variar el proceso
 - Menos intuitivo que variar el proceso directamente
- No se puede definir el *core process* de la línea de procesos
 - La variabilidad de SPEM se puede aplicar a cualquier parte de un método (y por tanto proceso)
 - No se puede establecer que parte de una línea de procesos es común para todos los procesos

52



Variabilidad en SPEM

Limitaciones a la variabilidad en SPEM

- No se puede evitar que una variación afecte a los objetivos del proceso
- Los elementos para introducir variabilidad son excesivamente complejos, y por tanto difícilmente reutilizables
- No existe una notación específica para variabilidad

53

Introducción

- Introducción
- Líneas de procesos software
 - Justificación y necesidad
- Modelado de Líneas de Proceso
 - SPEM
 - Descripción general
- Variabilidad en SPEM
- **Variabilidad en líneas de proceso software**

54

Mecanismos específicos de variabilidad en líneas de proceso

Nuevos mecanismos de variabilidad añadidos a SPEM

- LProcElement
 - Abstracción de todos los elementos derivados de las Líneas de Proceso Software
 - Superclase de *variant* and *Variation Point*

```

classDiagram
    class Classifier["InfrastructureLibrary::Core::Constructs::Classifier"]
    class LProcElement
    class VarPoint
    class Variant
    Classifier <|-- LProcElement
    LProcElement <|-- VarPoint
    LProcElement <|-- Variant
    VarPoint "1..*" -- "1..*" Variant : canOccupe
  
```

55

Mecanismos específicos de variabilidad en líneas de proceso

Nuevos mecanismos de variabilidad añadidos a SPEM

- Elementos *Variant* y *Variation Point* concretos
 - Estos elementos representan puntos de variación y variantes reales dentro de la línea de proceso
 - Heredan de las clases abstractas *variant* y *Variation Point*
 - Y también heredan de los elementos concretos de spem a los que les facilitan la capacidad de variar

```

classDiagram
    class VarPoint
    class Activity
    class Variant
    class VPActivity
    class VActivity
    VarPoint <|-- VPActivity
    Activity <|-- VPActivity
    Activity <|-- VActivity
    Variant <|-- VActivity
  
```

Mecanismos específicos de variabilidad en líneas de proceso

Nuevos mecanismos de variabilidad añadidos a SPEM

- Elementos *Variant* y *Variation Point* concretos
 - Se definen nuevos iconos para representar tanto a los puntos de variación como a las variantes

	Activity	Work ProductUs.	RoleUse.	TaskUse
Elemento SPEM				
VarPoint				
Variant				

Mecanismos específicos de variabilidad en líneas de proceso

Nuevos mecanismos de variabilidad añadidos a SPEM

- **Relación de *Occupation***
 - Hereda de la relación de generalización
 - Se define entre los puntos de variación y las variantes
 - Se han incluido restricciones para garantizar que sólo se pueda definir entre puntos de variación y variantes

```

classDiagram
    class InfraestructureLibrary::Core::Constructs::Generalization
    class Occupation
    class OccupationType
    class Opcional
    class Mandatory
    class Altemative["Altemative (-min: int, -max: int)"]
    OccupationType <|-- Occupation
    OccupationType <|-- Opcional
    OccupationType <|-- Mandatory
    OccupationType <|-- Altemative
    Occupation "1" -- "*" OccupationType : type
  
```

58

Mecanismos específicos de variabilidad en líneas de proceso

Nuevos mecanismos de variabilidad añadidos a SPEM

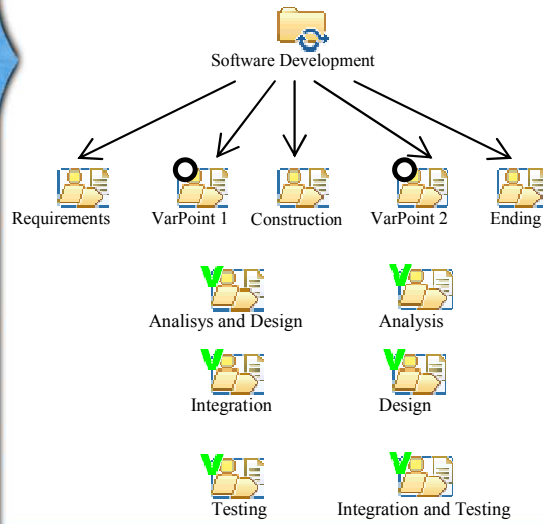
- **Dependencias**
 - Definidas entre los elementos de las líneas de proceso
 - Definición de restricciones para asegurar la integridad de los procesos
 - Diferentes tipos

```

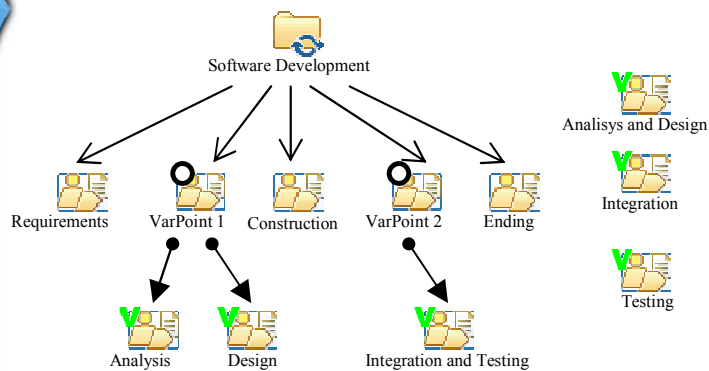
classDiagram
    class InfraestructureLibrary::Core::Constructs::Dependency
    class VariabilityDependencyType
    class VariabilityDependency
    class Inclusion
    class Exclusion
    class VariantToVariantDependency
    class VariantToVarPointDependency
    class VarPointToVarPointDependency
    Dependency <|-- VariabilityDependency
    VariabilityDependencyType "1" -- "0..*" VariabilityDependency : type of dependency
    VariabilityDependency "1" -- "0..*" VariabilityDependency : dependency
  
```

59

Línea de proceso

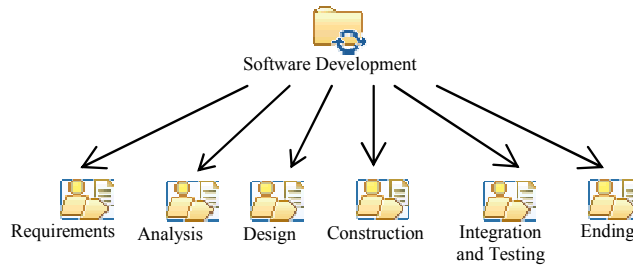


Línea de proceso con puntos de variación ocupados Como se ve

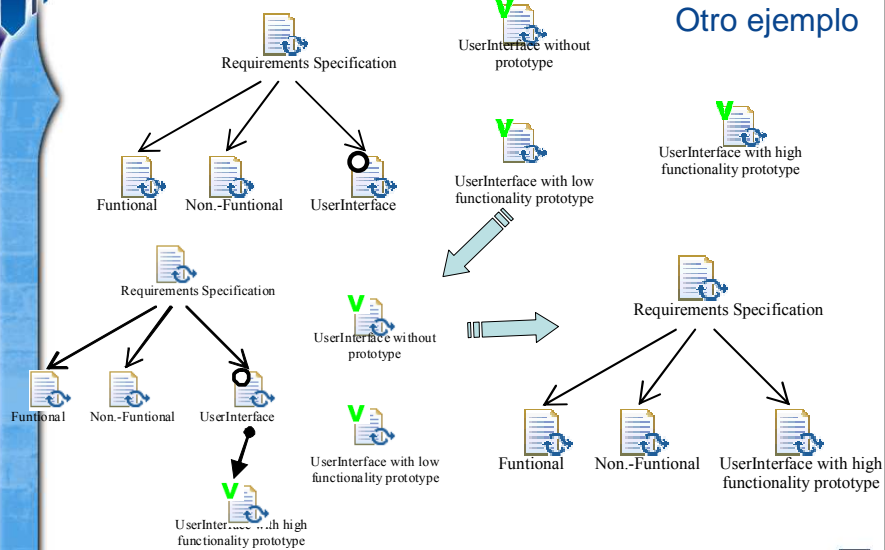


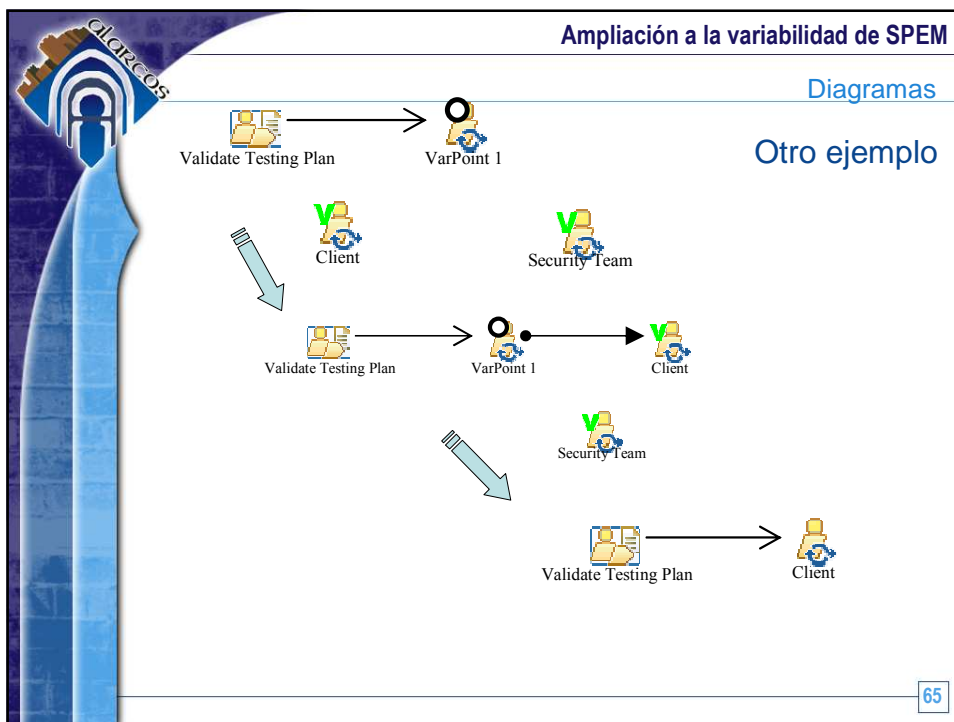
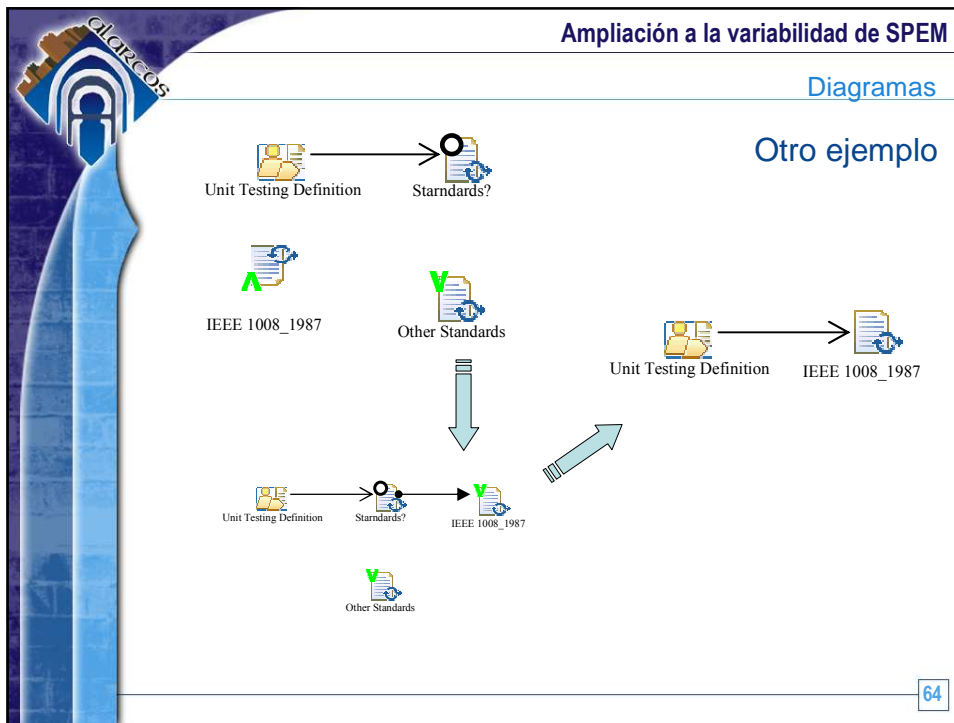
Visión del proceso generado

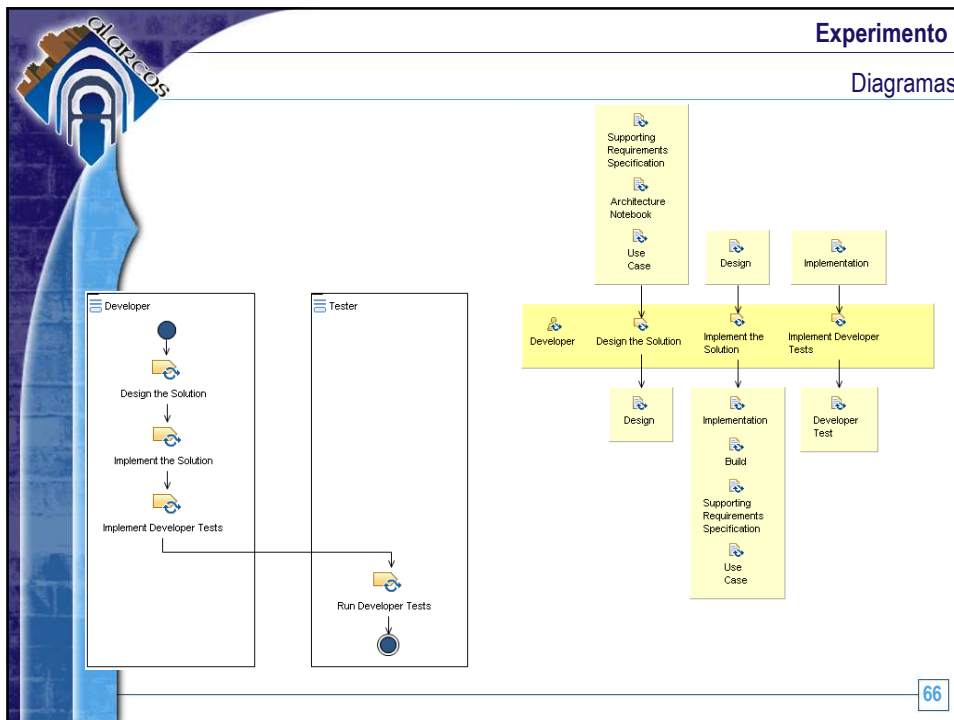
Lo que significa, una vez interpretadas las variantes dentro de los puntos de variación



Otro ejemplo







Referencias

OMG (2007). Software Process Engineering Metamodel Specification. Object Management Group.

Ruiz, F. and Verdugo, J. (2008). Guía de Uso de SPEM 2.0 con EPF Composer. Grupo Alarcos, Ciudad Real.

Martínez-Ruiz, T., García, F. and Piattini, M. (2008). Towards a SPEM v2.0 Extension to Define Process Lines Variability Mechanisms. In *Software Engineering Research, Management & Applications*, Vol. SCI 150 (Ed, Lee, R.) Springer Verlag. Praga, Czech Republic, pp. 115-130.

67