



## SCENARIO - DOMAIN ARCHITECT

### DOMAIN ARCHITECT

This perspective should develop and maintain the SPL architecture for all products in the company's portfolio. It must ensure that the component diagrams, by means of components and interfaces, incorporate the domain classes proposed and modeled in the class diagrams, thus showing the physical structure of the system as a whole, so that the components can then be rearranged according to SPL business rules and user requirements.

To diagrams correctly express the user requirements without inconsistencies, you must review such diagrams and elements. To achieve your goal, perform the steps outlined below to inspect each of the informed diagrams. When you find a defect in one of the steps, fill in the Defect Identification Form indicating the diagram, the step item (number question), and the element and the defect found.

### LOCATE THE CLASS DIAGRAM AND REQUIREMENTS SPECIFICATION

Step 1

inspection of class diagram

Step 2

### LOCATE THE COMPONENT AND CLASS DIAGRAM

Step 3

inspection of component diagram

Step 4



## LOCATE THE CLASS DIAGRAM AND REQUIREMENTS SPECIFICATION

The class diagram for your role should present the class structure and its relationships for all SPL systems in the project domain, paying attention to the application interfaces and their main methods. To ensure that this diagram describes the design classes in the correct way, carefully read the specification of requirements and while reading, make a list with the mentioned objects, the data that characterize this object and the possible methods. Compare the list made with the class diagram to ensure that there is no inconsistency between classes and user requirements. To do this, answer the questions that follow.

Step 1	For each class in the class diagram, check the classes, the attributes, the relationships between the classes with the list made and answer the questions that follow. Check the class after its analysis, to avoid being analyzed again.	
	1.1	Does the class name correctly express the objects of this class?
	1.2	Does this class correspond to an object that has not been defined in the requirements document? If so, please disregard it and their relationships to the next steps and go to the next class.
	1.3	Is this class in redundancy with another one already specified in the class diagram? If so, please disregard it and their relationships to the next steps and go to the next class.
	1.4	Is the class stereotyped in the class diagram?
	1.5	If the classes are grouped into packages, check and answer the questions that follow.
		1.5.1 Has the package name been defined and expresses the grouping correctly? If you have already filled out this defect for this package on the form you do not need to fill it out again
		1.5.2 Is the class in the correct package?
	1.6	For each relationship for this class, check the classes that make up this relationship, to ensure that the relationship between them is in accordance with the requirements specification. To do this, answer the questions that follow and mark the relationships already analyzed.
		1.6.1 Is this relationship in accordance with the requirements specification? Check if there is really a relationship between the elements for the context.
		1.6.2 Is the cardinality of this relationship correct according to the requirements specification?
		Check the type of relationship to ensure that the classes are stereotyped and correct according to the SMarty approach:
		• Generalization: Are the most general classifiers points of variation (<<variationPoint>>) and the most specific, variants?
		• Realization of interface: Are the specifications points of variation (<<variationPoint>>) and the implementations are variants?
		1.6.3 • Aggregation or Composition: Are the instances typed with diamonds (filled or not filled) are points of variation (<<variationPoint>>) and associated instances are variants?
		• Association: Check the AgregationKind attribute
		• if none: variants suggest to be mandatory (<<mandatory>>) or optional (<<optional>>). Check against the requirements specification. Is the stereotype correct?
		• whether the value * or 0..n are optional (<<optional>>)?
	1.6.4	Are classes that require another related to the <<requires>> stereotype?
	1.6.5	Are mutually exclusive classes related to the <<mutex>> stereotype?
	1.7	Was there any relationship missing for this class that was not specified in the class diagram?
	1.8	Interfaces specify methods that are externally visible to others. If the interface is of this type, analyze each of the defined methods and answer the questions that follow.



Step 2		1.8.1	Is the interface stereotyped with <<interface>>?
		1.8.2	Does the name of the method correctly express its function?
		1.8.3	Is the method redundant with another method previously defined for this class?
		1.9	Have the main methods been defined?
	After analyzing all the classes in the previous step (all marked as visited), check and analyze the questions that follow.		
	2.1	The control classes manage the activities of the class. For each class of this type that is stereotyped with <<optional>> and/or <<variationPoint>>. Check it to ensure that the variability/variant notations are correct according to the requirements specification. To do this, go to each of these classes and answer the questions that follow.	
		2.1.1	Is there a UML notation that represents variability (<<variability>>) associated with the control class?
		2.1.2	Have all the variants been defined and are they with the correct variant notation (<<OR >>, <<XOR>> or <<optional>>)?
	2.2	The <<variability>> stereotype represents variability through a UML comment. For each of these comments defined in the diagrams, go to the comment, analyze it and answer the questions below.	
		2.2.1	Are the variants defined in the variants set really variants for this element? If any are not, disregard it for the next questions.
		2.2.2	Are there any variants defined in the collection of variants that are not described in the class diagram?
		2.2.3	Are all variants related to this element defined with (<<OR>>), (<<XOR>>) or (<<optional>>) in the collection of variants of the meta-attribute variants <i>with</i> their correct name?
		2.2.4	Check the type of the associated variants:
• If they are of type <<optional>>, minSelection = 0 and maxSelection = 1?			
• If they are of type <<OR>>, minSelection = 1 and maxSelection = total of variants ?.			
		• If they are of type <<XOR>>, minSelection=maxSelection=1?	
2.3	Are there still items on your list that are not in any class? That is, it is missing from the class diagram (If they are variants that were already identified in the previous step, disregard).		



## LOCATE THE COMPONENT AND CLASS DIAGRAM

The component diagram in Domain Engineering should show the physical structure of the implementation through components, interfaces and their relationships to all system components, since some will be mandatory for all products and others will be arranged according to needs user-specific. To inspect it, make a list from the class diagram with all the classes in the system (remember to inspect the class diagram before). Compare the list with the component diagram to ensure that there is no inconsistency between them and that all classes are grouped into components. To do this, answer the questions that follow.

Step 3	Consider an element as a component or interface for the next steps. For each of them specified in the component diagram, check its correspondence with the list made, analyze it to answer each of the questions that follow. Remember to mark the elements of your list that have already been defined in any of the components.	
	3.1	Does the name correctly express the element?
	3.2	Is the element redundant with another one previously specified? If so, disregard it for the next steps and go to the next element
	3.3	Is the element on your list? If not, disregard it for the next steps and go to the next element.
	3.4	Is the element stereotyped in the component diagram?
	3.5	If the component is mandatory, is it marked with the <<mandatory>> stereotype?
	For each list of this element, check it and answer the questions that follow.	
	3.6	3.6.1 Does the component/interface relationship with another element really exist?
		3.6.2 If the relationship is of the type of contract, is the order provided/required or required/provided correct?
		3.6.3 Are variants that require the presence of another related to the <<requires>> or <<use>> stereotype?
		3.6.4 Are mutually exclusive variants related to the <<mutex>> stereotype?
	If the element is of the optional type, check it and answer the questions that follow.	
	3.7	3.7.1 Was the <<optional>> stereotype defined for the element?
		3.7.2 Is there a UML notation that represents variability (<<variability>>) associated with the element?
	If the element represents a variation point, check it and answer the questions that follow.	
	3.8	3.8.1 Was the <<variationPoint>> stereotype defined for the element?
		3.8.2 Is there a UML notation that represents variability (<<variability>>) associated with the element?
		3.8.3 Have all the variants been defined and are they with the correct variant notation (<<OR>> or <<XOR>>)?
		3.8.4 Are there any variants described in the component diagram that do not belong to this element?
		3.8.5 Is there a variant in redundancy with another one already specified?
		3.8.6 Is there a variant related to the element that was defined with (<<OR>>) or (<<XOR>>) that was not specified in the requirements? If so, remove it from the diagram and disregard this variant and its relationships for the next questions.
Step 4	After analyzing all the elements in the previous step (all marked as visited), check and analyze the questions that follow.	
	4.1	The <<variability>> stereotype represents variability through a UML comment. For each of these comments defined in the diagrams, go to the comment, analyze it and answer the questions below.



		4.1.1	Are the variants defined in the <i>variants</i> set really variants for this element? If any are not, disregard it for the next questions.	
		4.1.2	Are there variants defined in the collection of variants that are not described in the component diagram?	
		4.1.3	Are all variants related to this element defined with (<<OR>>), (<<XOR>>) or (<<optional>>) in the collection of variants of the meta-attribute <i>variants</i> <i>with</i> their correct name?	
		4.1.4	Check the type of the associated variants:	
			• If they are of type <<optional>>, minSelection = 0 and maxSelection = 1?	
	• If they are of type <<OR>>, minSelection = 1 and maxSelection = total of variants ?.			
	4.2		• If they are of type <<XOR>>, minSelection=maxSelection=1?	
			For components and interfaces described in the classifier format. Go to the classifier compartment and check the described operations.	
			4.2.1	Does the component/interface have the <<variationPoint>> stereotype defined?
			4.2.2	Are all component/port/interface variants to resolve this variation point visible in the operations compartment?
			4.2.3	Is there a port/component/interface specified in the operations compartment that does not belong to this element?
			4.2.4	Is there any redundancy for the ports/components/interfaces specified in the operations compartment?
	4.3		4.2.5	Are the stereotypes in the operations compartment correct according to each of the elements described there?
			Are there still items on your list that do not belong to any component? That is, it is missing from the component diagram (If there are variants that have already been identified in the previous step, disregard).	