



SCENARIO - DOMAIN DEVELOPER

DOMAIN DEVELOPER

This perspective should implement and test the components and interfaces that will be reusable throughout the product portfolio. You must develop according to the requirements of a range of products, for this you must ensure that the elements represented in the SMarty diagrams of class, sequence and component are not in disagreement with the requirements, contemplate the expected functions and allow you to have a vision implementation of reusable components.

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 CLASS DIAGRAM AND REQUIREMENTS SPECIFICATION

Step 1

inspection of use case diagram

Step 2

LOCATE THE COMPONENT AND CLASS DIAGRAM

Step 3

inspection of component diagram

Step 4

LOCATE THE SEQUENCE DIAGRAM, CLASS AND REQUIREMENT SPECIFICATION

Step 5

Step 6

Step 7

Step 8

Inspection of sequence diagram



LOCATE 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, including specific details for implementation, such as types of attributes and method parameters. 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 the classes and the user's requirements. To do this, answer the questions that follow.

Step 1	For each class in the class diagram, check the attributes, the relationships between the classes with the list made and answer the questions that follow. Make an appointment in 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, disregard her and her relationships for the next steps and move on 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 .
		• 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	Entity classes store data that identifies real-world concepts. If the data compartment and/or its types have been defined in the attribute compartment, analyze this information and for each attribute of this class



		answer the questions below.	
		1.8.1	Does the attribute name correctly express the real-world data?
		1.8.2	Does the attribute really belong to that class?
		1.8.3	Is the attribute redundant with another one already defined for this class?
		1.8.4	Is the attribute type correct?
		1.8.5	Is the visibility of the attribute in the class correct?
	1.9	Were the main attributes defined for this class?	
	1.10	The interfaces specify the methods externally visible to others. If the class is of this type, analyze each of the defined methods and answer the questions that follow.	
		1.10.1	Is the class stereotyped with <<interface>>?
		1.10.2	Does the name of the method correctly express its function?
		1.10.3	Is the method redundant with another method previously defined for this class?
		1.10.4	Check the input parameters for the method and answer the following questions:
			• Was any input parameter missing for the method?
			• Has a parameter been specified beyond what is necessary?
		1.10.5	Check the output parameters for the method and answer the following questions:
			• Is the return value correct?
	1.10.6	Is the visibility of the method correct?	
	1.11	Was there a lack of an important method for the implementation to be defined?	
	Step 2	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 <i>variants</i> 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 <i>variants</i> with 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 have already been 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?
	3.6	For each list of this element, check it and answer the questions that follow.
		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?
	3.7	If the element is of the optional type, check it and answer the questions that follow.
		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?
	3.8	If the element represents a variation point, check it and answer the questions that follow.
		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> with 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 ?.
			• If they are of type <<XOR>>, minSelection=maxSelection=1?
	4.2	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?
		4.2.5	Are the stereotypes in the operations compartment correct according to each of the elements described there?
	4.3	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).	



LOCATE THE SEQUENCE DIAGRAM, CLASS AND REQUIREMENT SPECIFICATION

The sequence diagram in Domain Engineering should express the interaction of the system: the exchange of messages between the objects of the systems that can be configured from an SPL from a technical point of view, close to the implementation that you must perform. With the sequence diagram in hand, check each of the described objects/actors and look for the corresponding class in the class diagram (if defined) or in the requirements specification. For each of them, check the relationships and messages that are exchanged between the elements to ensure that they are consistent with the class diagram/requirements specification. Then answer the questions that follow.

Step 5	Consider the "object heads" and the lifeline actors in the sequence diagram as an element. For each one of them, check the messages and stereotypes given to them in order to answer the questions that follow.	
	5.1	Does the name of the element correctly express the object/actor?
	5.2	Is the element represented in any class in the class diagram?
	5.3	Is the element part of this interaction according to the Requirements Specification? If not, disregard all the lifeline and their messages for the next steps and go to the next element.
	5.4	Is the element redundant with another defined element? If so, please disregard all the lifeline and their messages for the next steps and go to the next element.
	5.5	For each of the messages defined in the lifeline of this element, analyze it and answer the questions that follow.
		5.5.1 Is the message named?
		5.5.2 Is the interaction represented by this message described in the requirements specification? If not, disregard it and go to the next message.
		5.5.3 Is the order of the message correct according to the requirements specification?
		5.5.4 Does the message name match the method name in the class diagram?
		5.5.5 If specified, are the input parameters correct according to the class diagram?
		5.5.6 If specified, are the output parameters correct according to the class diagram?
	5.5.7	Messages that are not directly related to a variability and its elements, do not need a stereotype and are considered mandatory. Is the message of this type stereotyped?
	5.6	Have all messages for this element been defined in the sequence diagram according to the requirements specification and the class diagram?
	5.7	Are there redundant messages on this lifeline? If so, disregard it and move on to the next message.
	5.8	Are elements that require the presence of another related to the <<requires>> stereotype?
	5.9	Are mutually exclusive elements related to the <<mutex>> stereotype?
Step 6	For each alternative element in the sequence diagram such as the CombinedFragment with interactionOperator "alt" (alternative) and interactionUse "ref" element, check its stereotypes and messages related to this element to answer the questions that follow.	
	6.1	Is the element defined with the <<variationPoint>> stereotype?
	6.2	Is there a UML notation that represents variability (<<variability>>) associated with this/CombinedFragment element?
	6.3	Are the variants corresponding to the messages stereotyped correctly? Check the stereotypes of the variant messages for this variability.
		<ul style="list-style-type: none"> • For variants of the interactionUse element "ref": <<OR>> • For variants with interactionOperator "alt": <<XOR>>



Step 7	For each optional element in the sequence diagram, such as: combinedFragment with interactionOperator “opt” (optional) and exchange of messages between two non-mandatory objects or between one mandatory object and one not, check their stereotypes and messages related to it to answer the questions that follow.	
	7.1	Is there a UML notation that represents variability (<<variability>>) associated with this/CombinedFragment element?
	7.2	Are the variants corresponding to the/CombinedFragment messages correctly stereotyped with <<optional>>?
	7.3	For elements marked with <<optional>>, are the lifelines that are part of the CombinedFragment also stereotyped with <<optional>>?
Step 8	The <<variability>> stereotype represents variability through a UML comment. For each of these comments defined in the diagrams, review the questions below.	
	8.1	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.
	8.2	Do all variants in the variant set have an associated lifeline in the sequence diagram?
	8.3	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> with their correct name?
	8.4	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?	