Composite consistency rules

|  |  |  |
| --- | --- | --- |
| ID | The rule notation | Description |
| R1.5a | Xi«product»v«process»i«product» Ri«product»v«subprocess»i«product»  (acronym: XiviRivi) | Decomposition of the main process into sub-processes by products  For the rule set in Product one, create a Subprocess element with the name as in the Product element and copy the Product element to the Subprocess input. For the rule set in Product two, create a Subprocess element with the name as in the Product element and copy the Product element to the Subprocess output. For the rule set in Process, create a Subprocess element with the name as in the Process element, copy and connect the first one with the Product element to the subprocess input, and then copy and connect the second one with the Product element to the subprocess output. Filling in the parameter will create a Subprocess element with the name as in the parameter, while the first parameter should contain the name of the first object, and the last parameter should contain the name of the second object |
| R1.5.b | Xev«process»i«product» Rev«subprocess»i«product»  (acronym: XeviRevi) | Decomposition of the main process into Subprocesses by event and product  decomposition of the main process into Subprocesses by event and product  For a rule set in Event, create a Subprocess element with the name as in the Event element and copy the Event element to the Subprocess input (if they are connected), and the product in the Product element to the subprocess output, if they are connected. For a rule set in Product, create a Subprocess element with the name as in the Product element and copy the Product element to the Subprocess output, if they are connected, and copy the Event to the Subprocess input, if they are connected. For a rule set in Process, create a Subprocess element with the name as in the Process element, copy and connect the Event element to the Subprocess input, if they are connected, and then copy and connect the Product element to the Subprocess output, if they are connected. Filling in the parameter will create a Subprocess element with the name as in the parameter, while the first parameter should contain the event name, and the last parameter should contain the object name. If any elements are not connected, they will not be connected on the R diagram either. |
| R1.5.c | Xev«process»i«product» Ri«product»v«subprocess»i«product» (acronym: XeviRivi) | Decomposition of the main process into subprocesses by event and product into products and subprocesses  For a rule set in Event, create a Subprocess element with the name as in the Event element and create an Object element, with the type as the event name, at the Subprocess input (if they are connected), and an output in the Product element at the Subprocess output, if they are connected. For a rule set in Product, create a Subprocess element with the name as in the Product element and copy the Product element to the Subprocess output, if they are connected, and at the Subprocess input, create an object with the type as the event name, if they are connected. For a rule set in Process, create a Subprocess element with the name as in the Process element, copy and connect the Event element at the Subprocess input, if they are connected, and then copy and connect the Product element at the Subprocess output, if they are connected. Filling in the parameter will create a Subprocess element with the name as in the parameter, while the first parameter should contain the event name, and the last parameter should contain the object name. If any elements are not connected, they will not be connected on the R diagram either. |
| R1.10a | Xev«process»Bau  (acronym: XevBau) | Mapping an event to an actor and a process to a business Use Case  Create an Actor element with the same name as in the Event element and associate it with the created business Use Case with the same name as in the main process. |
| R1.10b | Xev«process»i«product»Baua  (acronym: XeviBaua) | Mapping an event and a product to the actors and the process to the business Use Case  Create an Actor element with the same name as in the Event element and associate it with the created business Use Case with the same name as in the main process. |
| R1.11a | Xv«process»i«product»Bau  (acronym: XviBau) | Mapping a product to an actor and a process to a business Use Case  Create an Actor element with the name as in the Product element and link it to the created business Use Case with the name as in the main process |
| R1.11b | Xi«product»v«process»i«product»  Baua  (acronym: XiviBaua) | Mapping products to actors and process to a business Use Case  Create a Use Case element named as in the main process, and then associate it with all created Actor elements named as all objects connected to the main process. |
| R1.12a | Xi«rules»v«process»Bau  (acronym: XivBau) | Mapping a rule to an actor and a process to a business Use Case  For the rule set in the Business Rule element, create an Actor element with the name as in the Business Rule element and associate it with the created business Use Case with the name as in the main process. |
| R1.19 | Rev«subprocess»Bau  (acronym: RevBau) | Map an event to an actor and a subprocess to a business Use Case  Creating an Actor element with the name as in the Event element and linking it to the created business Use Case with the name as in the subprocess.  If the rule is on an event, then the first found element of the related subprocess is retrieved. If the rule is on a subprocess, then the first found element of the related event is retrieved |
| R1.20 | Rev«subprocess»i«product»Baua  (acronym: ReviBaua) | Map the event and product to the actors and subprocess to the business Use Case  Creating the first Actor element with the name as in the Event element and the second element as in the object element and linking it to the created business Use Case with the name as in the subprocess.  If the rule is on an event, then the first found element of the related subprocess is retrieved, and then the first found element of the related object. If the rule is on a subprocess, then the first found element of the related event and the first found element of the related object are retrieved. If the rule is on an object, then the first found element of the related subprocess and the first found element of the related event are retrieved |
| R1.21 | Rv«subprocess»i«product»Bau  (acronym: RviBau) | Mapping a product to an actor and a subprocess to a business Use Case  Creating an Actor element named as in the object element and linking it to the created business Use Case named as in the subprocess.  If the rule is on an object, the first found element of the related subprocess is retrieved. If the rule is on a subprocess, the first found element of the related object is retrieved |
| R1.22 | Ri«product»v«subprocess»i«product»Baua  (acronym: RiviBaua) | Mapping products to actors and subprocesses to a business Use Case  Create a Use Case element named as in the subprocess, and then associate it with all created Actor elements named as all objects connected to this subprocess.  If the rule is on an object, then the first element found of the related subprocess is retrieved, and then all objects related to this found subprocess are searched. If the rule is on a subprocess, then all related objects are retrieved |
| R2.5a | Bau«scenarios»A(pv+)+  (acronym: BauA(pv+)+) | Mapping a business Use Case and its actors on the flow of activity control in a process diagram    Creation of Actions linked by Control Flow with the name as in the parameter ("v(1. Activity\_1)") where the names of the Actions must be placed in parentheses following the letter "v", and these constructs together create a chain of Partitions and Actions describing the scenario. In the absence of the parameter, a single action is created (if it does not already exist) with the name of the Use Case in the created Partition (if it does not exist) with the name of the first found Actor associated with the Use Case, if the rule is on the Use Case. If the rule is on the Actor element, the Partition has the name of the Actor, and the Action has the name of the first found Use Case associated with the Actor.  If the name of the pattern is placed in the Action parameter in curly brackets, then a diagram with the name of the pattern and elements of the pattern are created in the directory with the name of the Use Case (on which the rule is set). The Partition names of the pattern are replaced according to the parameters placed in the Partition parameter on the A side of the diagram, or (in the absence of parameters) according to the names of the Actors associated with the Use Case on which the rule is set. If the rule is set on Actor, the first found Use Case is selected as the main Subprocess (name of the diagram named Use Case). |
| R2.9a | Ri«subproduct»v«subprocess» i«subproduct»A(v+i+)+  (acronym: RiviA(v+i+)+) | Mapping a subprocess and its products (input and output) to the flow of data (objects) in the process diagram  Creation of ObjectFlow-related Instances named as in the parameter ("v(1. Activity 1)i(Instance1)”), where the Instance names must be placed in parentheses following the letter "i", and these constructs together create a chain of ObjectFlow-related Instances with appropriate Actions describing the scenario.  In the absence of parameters, a single action is created (if it does not already exist) with the name of the Subprocess with the ObjectFlow to the created Instance (if it does not exist) with the name of the first found Instance connected to the Subprocess, if the rule is on the Subprocess. If the rule is on the Instance element, the Instance has the name of the Instance (in the R diagram), and the Action has the name of the first found Subprocess connected to the given Instance.  If the name of the pattern is placed in the Action parameter in curly brackets, then a diagram with the name of the pattern and elements of the pattern are created in the directory with the name of the Subprocess (on which the rule is set). Input Instance names and output Instance names of the pattern are changed according to parameters placed in the Instance parameter on the diagram side A, or (in the absence of parameters) according to Input Instances and Output Instances connected to the Subprocess on which the rule is set. If the rule is set on Instance, the first found Subprocess is selected as the main Subprocess (name of the diagram named Subprocess). |
| R2.16a | Av«subprocess»Uau  (acronym: AvUau) | Mapping actions in a specific partition to a system Use Case linked to an actor  Create a Use Case with the name of the activity and connect to an Actor with the name of the partition the activity is in. Filling in the parameter for the "u" element will create a Use Case with the name of the parameter, and filling in the parameter for the "a" element will create an Actor with the name of the parameter. |
| R2.17a | ApUau | Mapping actions in a given partition to associations between the actor and system Use Cases  For the rule saved in the Partition element, an Actor element with the partition name will be created and a link to the Use Cases with the name of all activities that belong to the given partition in the process diagram will be created. Filling in the parameter in the "a" element will result in creating an Actor element with the name as in the parameter, while filling in the parameters in the "u" element will result in creating Use Case elements according to the order of occurrence of activities (actions) in the process diagram |
| R2.18a | Ai«product»v«subprocess»i«product»Cczc (acronym: AiviCczc) | Mapping Instances Related to Activities to Associative Classes  Creation of Class elements with the name equal to the Instance class name, if such a class does not already exist and association of these classes with the multiplicity of "1..\*" if the instances are connected via an action. |
| R2.22a | Ai«product»v«subprocess»i«product»Ccbh  (acronym: AiviCcbh) | Mapping instances associated to activities to associate classes and define class attributes and operations  Creation of Class elements with the name equal to the Instance class name, if such a class does not already exist and association of these classes with the multiplicity of "1..\*". Filling in the parameter will result in creation of the Class element, if such a class does not already exist, with the name as in the parameter. Depending on the content of the parameters in the "b" and "h" elements, creation of the given attributes and operations in the appropriate class. In the absence of parameters for the "h" elements, operations will be created with names like the names of the activities to which the instance corresponding to the given class is associated |
| R2.26a | Ai«product»«subprocess»Srt  (acronym: AivSrst) | Mapping all instance states to a state transition diagram in a specific state machine state region  Based on the instance class name, create a region and map the instance states, based on the connections of the given instance with the appropriate activities, to a state machine diagram |
| R3.5a | Uau«scenarios»Z(pv+)+  (acronym: UauZ(pv+)+) | Mapping a system Use Case and its actors to the flow of activity control in a system Use Case execution diagram  Creation of Actions linked by Control Flow with the name as in the parameter ("v(1.Activity\_1)v(2.Activity\_2)..."), where the names of the Actions must be placed in round brackets following the letter "v", and these constructions together create a chain of Actions describing the scenario. In the absence of the parameter, only two scenario nodes are created: Initial and ActivityFinal |
| R3.9a | CchZiv | Mapping class methods to class instances to instances entering and exiting mapped methods to activities  Creation of an Instance element with the name of the processed Class. Filling in the parameter will create an Instance with the name of the parameter. The rule is triggered either for all system Use Case realization diagrams, if there is no parameter or if there is no Use Case name after the double colon. If the Use Case name is identified, then the rule is triggered only for the system Use Case realization diagram corresponding to the specified Use Case. The double colon must always appear |
| R3.12b | SrsZ(v+i+)+ | Mapping the data flow taking into account the states of input objects and the states of output objects  Creation of Data Flow-related instances named as in the parameter ("v(1.Activity\_1) i(Instance\_1)v(2.Activity\_2)i(Instance\_2)..."), where the instance names must be placed in round brackets following the letter "i", and these constructs together create a chain of data flow-related instances with appropriate actions describing the scenario |
| R3.16a | ZpvYuUu | Mapping of the activities from the "System" partition to the intra-system Use Cases connected by the «include» relationship to the system Use Case for which the activity from the system Use Case execution diagram has been visualized  Creation of a Use Case (implementation Use Case) element with the name of the activity and connection to the Use Case (system Use Case) element for which a specific activity has been visualized. Filling in the parameter for the "u" element on the Y diagram side will create a Use Case (implementation Use Case) element with the name as in the parameter, and filling in the parameter for the "u" element on the U diagram side will create a Use Case (system Use Case) element with the name as in the parameter |
| R3.21a | ZiviJcbh | Mapping associated instances to activities to associative classes, and define class attributes and class operations based on dataflow-related activities with instances  Creation of Class elements with the name equal to the Instance class name, if such a class does not already exist and association of these classes with the multiplicity of "1..\*". Filling in the parameter will result in creation of the Class element, if such a class does not already exist, with the name as in the parameter. Depending on the content of the parameters in the "b" and "h" elements, creation of the given attributes and operations in the appropriate class. In the absence of parameters for the "h" elements, operations will be created with names like the names of the activities to which the instance corresponding to the given class is associated |
| R3.25a | ZivTrst | Mapping all instance states to a state transition diagram in a specific region of the system state machine  Based on the instance class name, create a region and map the instance states, based on the connections of the given instance with the appropriate activities, to a state machine diagram |
| R4.6a | YuQ(lm+)+ | Mapping the steps of the scenario to the transmission of messages between Lifelines  Creation of Messages (and Responses) linked by the Execution element (UML Occurrence) between the Lifeline elements listed in the parameter in the format "message\_name\_1(Lifeln\_1,Lifeln\_2); message\_name\_3(Lifeln\_3);...", where the names of the Lifeline elements must be placed in round brackets following the Message name. For the first Message, two Lifeline elements must be listed separated by a comma, while the next Messages require only one Lifeline element. Constructs with a message must be separated by a semicolon. The rule is not triggered in the absence of a parameter. |
| R4.10a | JchQlm+l | Mapping class operations to message exchange between lifelines  Create multiple Messages (and Responses) between Lifeline elements (generated from the class) according to methods placed on the class. The order in which the Messages are sent can be specified in the message parameter. |
| R4.12a | TrstsQlsms | Mapping the system states of the machine to the states of the selected lifeline between the sending and receiving of a message with a specific name  Creation of Data Flow-related instances named as in the parameter ("v(1.Activity\_1) i(Instance\_1)v(2.Activity\_2)i(Instance\_2)..."), where the instance names must be placed in round brackets following the letter "i", and these constructs together create a chain of data flow-related instances with appropriate actions describing the scenario |
| R4.16a | QlmlM(qyq)+ | Mapping Lifelines and messages to components and interfaces between them  Creation of an Interface element, if it does not already exist, with the name of the Message element between the Components resulting from the Lifeline element mapping between which this Message is defined. Filling in the parameter will create an additional Component element with the name as in the parameter, which will mediate between the Components resulting from the Lifeline element mapping between which this Message is defined. In the latter case, Interfaces will not be created between the additional component, only if there is an Interface between the Component resulting from the target Lifeline element mapping to which this Message is sent |