Simple consistency rules

|  |  |  |
| --- | --- | --- |
| ID | The rule notation | Description |
| R1.1 | XeRi«information» | generating process “information” instances from events  Creation of an Information element with the same name as the Event element. Filling in the parameter will create an Information element with the same name as in the parameter |
| R1.2 | Xev«process»Rv«subprocess» | generating sub processes from events and the main process  For a rule set in Event, the creation of a Subprocess element with the same name as the Event element. For a rule set in Process, the creation of a Subprocess element with the same name as the Process element. Filling in the parameter will create a Subprocess element with the same name as in the parameter |
| R1.3 | Xi«product»Ri«subproduct» | generating “subproduct” instances from context “product” instances  For a rule set in Object, creating an Object element with the name as in the mapped Object element. Filling in the parameter will create an Object element with the name as in the parameter |
| R1.4 | Xi«rules»v«process» Rv«subprocess» | generating sub processes from “rules” instances and the main process  For a rule set in Business Rule, create a Subprocess element with the name as in the Business Rule element. For a rule set in Process, create a Subprocess element with the name as in the Process element. Filling in the parameter will create a Subprocess element with the name as in the parameter |
| R1.5 | Xv«process»i«product» Rv«subprocess» | generating sub processes from the main process and “product” instances  For a rule set in Product, create a Subprocess element with the name as in the Product element. For a rule set in Process, create a Subprocess element with the name as in the Process element. Filling in the parameter will create a Subprocess element with the name as in the parameter |
| R1.6 | Xz«control»Rz«data» | generating process object flows from control flows  Creating a DataFlow between the Information element and the Subprocess element. Filling in the parameter will give the DataFlow the name as in the parameter |
| R1.7 | Xz«data»Rz«data» | generating process object flows from context object flows  Creating a DataFlow between a Subprocess element and a Subproduct, Business Rule, or Object element. Filling in the parameter will give the DataFlow the name as in the parameter |
| R1.8 | XiSTATERiSTATE | generating process inState\_objectNodes from context inState\_objectNodes  Setting the Instance state to a previously created Subproduct element (of this Product), named as the Product state. Filling in the parameter will create a Subproduct element named as in the parameter ("Subproduct[State]") and set the Product state to the name as in the parameter ("Subproduct[State]"). |
| R1.9 | XeBu | generating business Use Cases from events  Create a Use Case element with the name as in the Event element. Filling in the parameter will create a Use Case element with the name as in the parameter. |
| R1.10 | XeBa | generating business actors from events  Create an Actor element with the same name as in the Event element. Filling in the parameter will create an Actor element with the same name as in the parameter |
| R1.11 | Xi«product»Ba | generating business actors from context “product” instances  Create an Actor element with the name as in the Product element. Filling in the parameter will create an Actor element with the name as in the parameter. |
| R1.12 | Xi«rules»v«process»Bu | generating business Use Cases from “rules” instances and the main process  For a rule set in the Business Rule element, a Use Case element will be created with the name as in the Business Rule element. For a rule set in the Process element, a Use Case element will be created with the name as in the Process element. Filling in the parameter will create a Use Case element with the name as in the parameter. |
| R1.13 | Xei«rules»v«process» Bu«scenarios» | generating business Use Cases scenarios from events, “rules” instances and the main process  In the current version of the tool, the functionality of mapping the process to use cases in terms of the scenario has not been implemented. |
| R1.14 | Xi«rules»v«process»Ba | generating business actors from “rules” instances and the main process  For a rule set in the Business Rule element, an Actor element will be created with the name as in the Business Rule element. For a rule set in the Process, an Actor element will be created with the name as in the Process element. Filling in the parameter will create an Actor element with the name as in the parameter. |
| R1.15 | Xz«control»Bz«func» | generating business Use Case associations from context control flows  Creating an Association between the Actor element and the Use Case element. Filling in the parameter will name the Association as in the parameter |
| R1.16 | Xv«data»Bu«data» | generating data part of activities from data part of business Use Cases  In the current version of the tool, the functionality of mapping the process to Use Cases in terms of the scenario has not been implemented. |
| R1.17 | Xv«process»i«product»Bu | generating business Use Cases from the main process and “product” instances  For a rule set in Product, create a Use Case element with the name as in the Product element. For a rule set in Process, create a Use Case element with the name as in the Process element. Filling in the parameter will create a Use Case element with the name as in the parameter. |
| R1.18 | Xz«data»Bz«func» | generating business Use Case associations from data flows  Create an Association between the Actor element and the Use Case element. Filling in the parameter will name the Association as in the parameter |
| R2.1 | BaApH | generating horizontal partitions from business actors  Creating a Partition element with the name as in the Actor element. Filling in the parameter will create a Partition element with the name as in the parameter |
| R2.2 | Bu«data»Ai | generating business instances from data part of business Use Cases  In the current version of the tool, the functionality of mapping Use Case data description to instances has not been implemented |
| R2.3 | Bu«data»Av«data» | generating data part of activities from data part of business Use Cases  In the current version of the tool, the functionality of mapping Use Case data description to instances has not been implemented |
| R2.4 | Bu«states»AiSTATE | generating business inState\_objectNodes from a part of business Use Case state  Creating an Instance element with the name as in the parameter ("Instance[State]") and setting the Instance state with the name as in the parameter ("Instance[State]"). Instances with their states must be separated by semicolons. The absence of the parameter will not trigger the rule. The current version of the tool does not implement the functionality of binding Instances to each other |
| R2.5 | Bu«scenarios» An«start»(v+|i+)+n«stop» | generating business Use Cases realization diagram from business Use Case scenarios  Creation of Actions related to 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. The current version of the tool does not implement the functionality of creating Instances and binding them to Actions |
| R2.6 | Bz«func»ApHv | mapping business activities to horizontal partitions from business Use Case associations  Assignment of the Actions listed in the parameter to the Partition that was previously created based on the execution of the R2.1 consistency rule and corresponding to the Actor with which the processed Association is related. The parts of the Action name listed in the parameter must be separated by a semicolon. The assignment is made after determining whether the listed fragments of strings are included in the name of the previously created Actions. The absence of the parameter will not cause the rule to be executed |
| R2.7 | Bu«data»Az«data» | generating business data flows from a part of business Use Case data  In the current version of the tool, the functionality of mapping the Use Case data description to the data flow has not been implemented |
| R2.8 | Ri«information»ApV | generating business vertical partitions from process “information” instances  Creation of a Partition element with the name as in the Information element. Filling in the parameter will create a Partition element with the name as in the parameter. The information must have a DataFlow |
| R2.9 | Ri«information»ApH | generating business horizontal partitions from process “information” instances  Creation of a Partition element with the name as in the Information element. Filling in the parameter will create a Partition element with the name as in the parameter. The information must have a DataFlow |
| R2.10 | Ri«subproduct»ApV | generating business vertical partitions from process “subproduct” instances  Creation of a Partition element with the name as in the Subproduct element. Filling in the parameter will create a Partition element with the name as in the parameter. The subproduct must have a DataFlow |
| R2.11 | RiSTATEAiSTATE | generating business inState\_objectNodes from process inState\_objectNodes  For a rule set in Information, create an Instance element with the name as in the Information element and set the Instance state to the name as in the parameter (“[State]”). The states must be separated by semicolons. Filling the parameter with the names of the Instances and their states (“Instance[State];Instance[State]…”) will create Instance elements with their states named as in the parameter. The absence of the parameter will not cause the rule to run. For a rule set in Subproduct, create an Instance element with the name as in the Subproduct element and set the Instance state to the name as in the Subproduct. Filling the parameter with the names of the Instances and their states (“Instance[State];Instance[State]…”) will create Instance elements with their states named as in the parameter |
| R2.12 | Rz«data»Az«data» | generating business data flows from process data flows  In the current version of the tool, the functionality of mapping relationships between instances (product) to relationships between instances has not been implemented |
| R2.13 | Ri«information»Ai | generating business instances from process „information” instances  Creation of an Instance element with the name as in the Information element. Filling in the parameter will create an Instance element with the name as in the parameter. The information must have a DataFlow |
| R2.14 | RiAi | generating business instances from process instances  Create an Instance element with the name as in the Subproduct element. Filling in the parameter will create an Instance element with the name as in the parameter. The subproduct must have a DataFlow. |
| R2.15 | ApHUa | generating system actors from business horizontal partitions  Creating an Actor element with the name Partition. Filling in the parameter will create an Actor element with the name as in the parameter |
| R2.16 | AvUu | generating system Use Cases from business activities  Creation of a Use Case element with the name of the activity. Filling in the parameter will create a Use Case element with the name as in the parameter |
| R2.17 | ApHvUz«func» | generating system Use Case associations from business activities in horizontal partitions  Creation of an Association element with the name of Partition. Filling in the parameter will create an Association element named as in the parameter |
| R2.18 | AiCc | generating business classes from business instances  Creating a Class element with the name equal to the Instance class name, if such a class does not already exist. Filling in the parameter will create a Class element, if such a class does not already exist, with the name as in the parameter |
| R2.19 | Az«data»Cz | generating business class associations from business data flows  Creating an Association element between classes that were created from mapping Instances connected by a data flow with this rule set. Filling the parameter in the data flow attached to the Instance will set the Association multiplicity with the content as in the parameter. Lack of data flow between Instances will not trigger the rules set on the flows connecting them |
| R2.20 | AvCch | generating business class operations from business activities  Creating an Operation element in a created Class (e.g., in the R2.18 rule on a given Instance) with the name as in the parameter ("Class::name:type"), if this Operation has not been created before. The absence of a parameter will not trigger the rule. Operations must be separated by semicolons |
| R2.21 | Av«data»Ccb | generating business class attributes from a part of business activity data  Creating an Attribute element in a created Class (e.g. in the R2.18 rule on a given Instance) with the name as in the parameter ("Class::name:type"), if this Attribute has not been created before. The absence of the parameter will not trigger the rule. Attributes must be separated by semicolons |
| R2.22 | AiCcb | generating business class attributes from slots of business instances  Creating an Attribute element in a previously created Class (in a previously set R2.18 rule on a given Instance) with the name as in the parameter ("name:type"), if this Attribute has not been created before. The absence of the parameter will not trigger the rule. Attributes must be separated by semicolons |
| R2.23 | AiSTATESs | generating business states from business inState\_objectNodes  Creation of a State element in a previously created Region (in a previously set R2.25 rule on a given Instance) with the same name as the Instance state, if this element has not been created before. The parameter is not processed |
| R2.24 | Az«data»St | generating business transitions from business data flows  Create a Transition element, if one does not already exist, between previously existing States that are created from the Instance mapping. Data flows connecting these Instances must have this rule set. The parameter is not processed |
| R2.25 | AiSr | generating business regions from business instances  Create a Region element with the name of the Instance, if it does not already exist, and a Pseudostate Initial and Final. Filling in the parameter will create a Region element with the name as in the parameter |
| R2.26 | AvSt | generating business transitions from business activities  In the current version of the tool, the functionality of mapping activities to state transitions has not been implemented |
| R3.1 | UaZpV | generating system vertical partitions from system actors  Creating a Partition element with the name as in the Actor element. Filling in the parameter will create a Partition element with the name as in the parameter |
| R3.2 | Uu«data»Zi | generating system instances from a part of system Use Case data  In the current version of the tool, the functionality of mapping Use Case data description to instances has not been implemented. |
| R3.3 | Uu«data»Zv«data» | generating a part of system activities data from a part of system Use Case data  In the current version of the tool, the functionality of mapping Use Case data description to instances has not been implemented. |
| R3.4 | Uu«states»ZiSTATE | generating system inState\_objectNodes from a part of system Use Case state  Creating an Instance element with the name as in the parameter ("Instance[State]") and setting the Instance state with the name as in the parameter ("Instance[State]"). Instances with their states must be separated by semicolons. The absence of the parameter will not trigger the rule. The current version of the tool does not implement the functionality of binding Instances to each other |
| R3.5 | Uu«scenarios»Zn«start»?(v+|i+)+n«stop»? | generating system Use Cases realization diagram from system Use Case scenarios  Creation of Actions connected with 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 structures together create a chain of Actions describing the scenario. In the absence of the parameter, two nodes are created: Initial and ActivityFinal. The current version of the tool does not implement the functionality of creating Instances and binding them to Actions |
| R3.6 | Uz«func»ZpVv | mapping system activities to vertical partitions from system Use Case associations  Assignment of the Actions listed in the parameter to the Partition that was previously created based on the execution of the R2.1 consistency rule and corresponding to the Actor with which the processed Association is related. The parts of the Action name listed in the parameter must be separated by a semicolon. The assignment is made after determining that the listed fragments of strings are included in the Action name. The lack of the parameter will not cause the rule to be executed |
| R3.7 | Uu«data»Zz«data» | generating system data flows from a part of system Use Case data  In the current version of the tool, the functionality of mapping the Use Case data description to the data flow has not been implemented |
| R3.8 | CzZz«data» | generating system data flows from business class associations  In the current version of the tool, the functionality of mapping the Use Case data description to the data flow has not been implemented |
| R3.9 | CcZi | generating system instances from business classes  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 instances on the system Use Case realization diagram, 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 |
| R3.10 | CchZv | generating system activities from business class operations  Creation of an Action element with the name as in the Class Operation element. Filling in the parameter will create an Action element with the name as in 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 a 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 |
| R3.11 | CcbZv«data» | generating a part of system activity data from business class attributes  In the current version of the tool, the functionality of mapping class attributes to the description of given activities has not been implemented |
| R3.12 | SsZiSTATE | generating system inState\_objectNodes from business states  Creating an Instance element named as Region, which contains the processed State and setting the Instance state to the name of the processed State. Filling in the parameter (“[State\_1];[State\_2]::UCname”) will create an Instance named as Region, which contains the processed State and setting the state of the next Instances named State\_1, State\_2 in the parameter, if these names are entered. Otherwise, the state name is taken from the processed State. 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. State names in the parameter must be separated by semicolons, if they appear. A double colon must always appear |
| R3.13 | SrZi | generating system instances from business regions  Create an Instance element with the name of the Region being processed. Filling in the parameter will create an Instance with the name of the parameter |
| R3.14 | StZz«data» | generating system data flows from business transitions  In the current version of the tool, the functionality of mapping the transition of the business state machine to the data flow has not been implemented |
| R3.15 | ZpVYa | generating internal actors from system vertical partitions  Creating an Actor element with the name of the Partition. Filling in the parameter will create an Actor element with the name as in the parameter |
| R3.16 | ZvYu | generating internal Use Cases from system activities  Creation of a Use Case element with the name of the activity. Filling in the parameter will create a Use Case element with the name as in the parameter |
| R3.17 | ZpVvYz«func» | generating internal Use Case associations from system activities in vertical partitions  Creation of an Association element named as the Partition. Filling in the parameter will create an Association element named as in the parameter |
| R3.18 | ZiJc | generating system classes from system instances  Creating a Class element, if it does not already exist, with the name equal to the Instance class name. Filling in the parameter will create a Class element, if such a class does not already exist, with the name as in the parameter |
| R3.19 | Zz«data»Jz | generating system class associations from system data flows  Creating an Association element between classes that were created from mapping Instances connected by a data flow with this rule set. Filling the parameter in the data flow attached to the Instance will set the Association multiplicity with the content as in the parameter. Lack of data flow between Instances will not trigger the rules set on the flows connecting them |
| R3.20 | ZvJch | generating system class operations from system activities  Creation of an Operation element, if it does not already exist, in the created Class (e.g. in the R2.18 rule on the given Instance) with the name as in the parameter ("Class::name:type"). The absence of the parameter will not trigger the rule. Operations must be separated by semicolons |
| R3.21 | Zv«data»Jcb | generating system class attributes from a part of system activity data  Creating an Attribute element, if it does not already exist, in the created Class (e.g. in the R2.18 rule on the given Instance) with the name as in the parameter ("Class::name:type"). The absence of the parameter will not trigger the rule. Attributes must be separated by semicolons |
| R3.22 | ZiSTATETs | generating system states from system inState\_objectNodes  Creation of an Attribute element, if it does not already exist, in a previously created Class (in a previously set R2.18 rule on a given Instance) with the name as in the parameter ("name:type"). The absence of a parameter will not trigger the rule. Attributes must be separated by semicolons |
| R3.23 | Zz«data»Tt | generating system transitions from system data flows  Creation of a State element in a previously created Region (in a previously set R2.25 rule on a given Instance), if this element has not been created before, with the same name as the Instance state. The parameter is not processed |
| R3.24 | ZiTr | generating system regions from system instances  Create a Transition, if it does not already exist, between previously existing States that were created from the Instance mapping. Data flows connecting these Instances must have this rule set. The parameter is not processed |
| R3.25 | ZvTt | generating system transitions from system activities  Create a Region element, if it does not already exist, with the name of the Instance, and the Pseudostates Initial and Final. Filling in the parameter will create a Region element with the name as in the parameter |
| R4.1 | YaQl | generating lifelines from internal actors  Create a Lifeline element, if it does not already exist, with the name of the Actor. Filling the parameter will create a Lifeline element with the name as in the parameter |
| R4.2 | YuQl | generating lifelines from internal Use Cases  Create a Lifeline element, if it does not already exist, with the name of the Use Case. Filling in the parameter will create a Lifeline element with the name as in the parameter |
| R4.3 | Yu«data»Qm«parameter» | generating message parameters from a part of internal Use Case data  In the current version of the tool, the functionality of mapping the data description to the message signature has not been implemented |
| R4.4 | Yu«states»Qs | generating sequence states from a part of internal use case states  In the current version of the tool, the functionality of mapping the list of instance states to state Invariants elements has not been implemented |
| R4.5 | Yu«states»Qo | generating sequence occurrences from a part of internal Use Case states  In the current version of the tool, the functionality of mapping the list of instance states to execution elements has not been implemented |
| R4.6 | Yu«scenarios»Qm | generating messages from internal Use Case scenarios  Creation of Messages (and Responses) linked by the Execution element (UML Occurence) 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.  If the next rule for the element being processed is the rule "Yu«data»Qm«parameter»" (R4.3), then the signature of the Message (and Response) is filled in according to the content of this rule |
| R4.7 | Yz«func»Qm | generating messages from internal Use Case associations  In the current version of the tool, the functionality of mapping associations between an actor and a Use Case to a message has not been implemented. |
| R4.8 | JcQl | generating lifelines from system classes  Creation of a Lifeline element, if it does not already exist, named as the Class. Filling in the parameter will create a Lifeline element with the name as in the parameter.  The rule is triggered either for all realization diagrams inside the system Use Case, if there is no parameter, or if there is no Use Case name after a double colon in the parameter. If a Use Case name is identified, then the rule is triggered only for the realization diagram inside the system Use Case corresponding to the specified Use Case. The double colon must always appear |
| R4.9 | JchQm | generating messages from system class operations  Creation of a Message (and Response) with the signature of the Operation on the created Execution (UML Occurence) on the target Lifeline element with the format "message\_name(Lifeln\_1,Lifeln\_2)::use\_case\_name", whereby the names of Lifeline elements must be placed in round brackets following the Message name. In the case of existence of such a Message (and Response), the execution of the consistency rule results in supplementing the Message (and Response) with the Operation call signature. In the case when the source and target Lifeline elements are identical, a self-call is created (R.4.10 JzQm«self»). The rule is not executed in the case of the missing parameter.  The rule is executed either for all implementation diagrams within the system Use Case, when there is no parameter or there is no Use Case name after a double colon in the parameter. If a Use Case name is identified, then the rule is triggered only for the Use Case realization diagram within the system Use Case corresponding to the named Use Case |
| R4.10 | JcbQm«parameter» | generating message parameters from system class attributes  In the current version of the tool, the functionality of mapping class attributes to message signature has not been implemented |
| R4.11 | JzQm«self» | generating self-messages from system class associations  In the current version of the tool, the functionality of mapping associations between classes to self-calls has not been implemented. This rule is replaced by the rule R4.9 JchQm for identical Lifeline elements |
| R4.12 | TsQo | generating sequence occurrences from system states  Creation of a Message (and Response) on the created Execution (UML Occurrence) on the target Lifeline element with the format "message\_name(Lifeln\_1,Lifeln\_2)::use\_case\_name", where the names of Lifeline elements must be placed in round brackets following the Message name. The rule is not triggered if the parameter is missing.  The rule is triggered either for all realization diagrams inside the system Use Case, if the parameter is missing, or if the Use Case name is missing after a double colon in the parameter. If the Use Case name is identified, then the rule is triggered only for the realization diagram inside the system Use Case corresponding to the specified Use Case |
| R4.13 | TtQo | generating sequence occurrences from system transitions  Creation of a Message (and Response) on the created Execution (UML Occurrence) on the target Lifeline element with the format "message\_name(Lifeln\_1,Lifeln\_2)::use\_case\_name", where the names of Lifeline elements must be placed in round brackets following the Message name. The rule is not triggered if the parameter is missing.  The rule is triggered either for all realization diagrams inside the system Use Case, if the parameter is missing, or if the use case name is missing after a double colon in the parameter. If the Use Case name is identified, then the rule is triggered only for the realization diagram inside the system use case corresponding to the specified Use Case |
| R4.14 | TrQl | generating lifelines from system regions  Creation of a Lifeline element, if it does not already exist, with the name of Region. Filling in the parameter will create a Lifeline element with the name as in the parameter.  The rule is triggered either for all realization diagrams inside the system Use Case, if there is no parameter or if there is no Use Case name after a double colon in the parameter. If a Use Case name is identified, then the rule is triggered only for the realization diagram inside the system Use Case corresponding to the specified Use Case. The double colon must always appear |
| R4.15 | QlMq | generating components from lifelines  Create a Component element, if it does not already exist, with the name of the Lifeline element. Filling in the parameter will create a Component element with the name as in the parameter |
| R4.16 | QmMy | generating interfaces from messages  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 |
| R4.17 | Qm«parameter»My«data» | generating interface parameters from messages parameters  In the current version of the tool, the functionality of mapping message signatures to interface parameters has not been implemented |
| R4.18 | QoMy«control» | generating interface behavior from sequence occurrences  In the current version of the tool, the functionality of mapping calls to interface specifications has not been implemented |