Property Sheet Simplification II

This document is a continuation of a previous proposal for simplification of the **Properties View** content in the UML Light project, covering the remaining diagram types *State Machine Diagram* and *Activity Diagram*. As before, the focus is primarily on the **UML** tab, the first tab presented for each element and also the default. For anything not presented on this tab, there is always the **Advanced** tab that presents all attributes and the **Model Explorer** for management of owned elements.

State Machine Concepts

State Machine

Unsupported properties to be hidden:

- isActive — as for `Class` (from which this is inherited)
- isReentrant
- use case — only classes are supported as subjects in UML Light

Oddly, this property sheet does not present the standard `Behavior` constraints (`postcondition`, `postcondition`, `bodyCondition`) that are show in the **UML** tab for `Interaction`. The same **Constraints** tab as proposed for the `Interaction` should be shown for `StateMachine` also, or else it should be omitted for `Interaction`.

State Machine

Region

No changes (only the name is currently shown, plus the localized label).

*It may be worth adding the region entirely in both the **Model Explorer** view (using its facet customization capability) and in the diagram (presenting the state machine properties when the region is selected) as the definition of multiple regions, and especially composite states, is not supported in UML Light.*

Initial Pseudostate

No changes (the `kind` attribute is read-only and useful).

Choice Pseudostate

No changes (the `kind` attribute is read-only and useful).

Junction Pseudostate

No changes (the `kind` attribute is read-only and useful).

State, FinalState

Unsupported properties to be hidden:

- submachine — connection points and state machine decomposition are not supported in *UML Light*
- deferrable triggers
- state invariant — the **New Child** menu is required to exclude `Constraint` on state machine elements, including `State`, so it doesn't make sense to present this here
- entry/doActivity/exit — the **New Child** menu is required to exclude `Behavior` types on state machine elements, including `State`, so it doesn't make sense to present these here

Excluding the activities implies that the *Xtext*-based editor for states should also be disabled, as this allows for a textual specification of these elements.

State

Transition

Unsupported properties to be hidden:

- kind — the distinction between internal and external transitions is not supported in *UML Light*
- triggers
- guard — the **New Child** menu is required to exclude `Constraint` on state machine elements, including `Transition`, so it doesn't make sense to present this here
- effect — the **New Child** menu is required to exclude `Behavior` types on state machine elements, including `Transition`, so it doesn't make sense to present these here

Excluding the guard and effect implies that the *Xtext*-based editor for transitions should also be disabled, as this allows for a textual specification of these elements.

Transition

Activity Concepts

Activity

Unsupported properties to be hidden:

- isActive — as for `Class` (from which this is inherited)
- isReentrant
- isSingleExecution
- precondition, postcondition — as for `Interaction` should be moved to a **Constraints** tab of their own, or else just omitted as decided in that case
- owned parameter — not supported for creation in *UML Light*
- variable — not supported for creation in *UML Light*

Excluding the activities implies that the *Xtext*-based editor for states should also be disabled, as this allows for a textual specification of these elements.

Activity

InitialNode, ForkNode

No changes

FlowFinalNode, ActivityFinalNode, MergeNode

Unsupported properties to be hidden:

- visibility — not interesting in a *UML Light* context and not shown on other nodes

FlowFinalNode

DecisionNode

Unsupported properties to be hidden:

- visibility — not interesting in a *UML Light* context and not shown on other nodes
- decisionInput — the **New Child** menu is required to exclude `Behavior` types on activity elements, including `DecisionNode`, so it doesn't make sense to present this here
- decisionInputFlow — the diagram is required to exclude `ObjectFlow`, so it doesn't make sense to present this here

DecisionNode

JoinNode

Unsupported properties to be hidden:

- joinSpec — the **New Child** menu is required to exclude `ValueSpecification` types on activity elements, including `JoinNode`, so it doesn't make sense to present this here

JoinNode

OpaqueAction

Unsupported properties to be hidden:

- visibility — not interesting in a *UML Light* context and not shown on other nodes
- local precondition/postcondition — the **New Child** menu is required to exclude `Constraint` types on activity elements, including `OpaqueAction`, so it doesn't make sense to present these here. Or, they could be moved to a **Constraints** tab as for `Interaction` and other behaviors

OpaqueAction

CallOperationAction

Unsupported properties to be hidden:

- visibility — not interesting in a *UML Light* context and not shown on other nodes
- local precondition/postcondition — the **New Child** menu is required to exclude `Constraint` types on activity elements, including `CallOperationAction`, so it doesn't make sense to present these here. Or, they could be moved to a **Constraints** tab as for `Interaction` and other behaviors

CallOperationAction

CallBehaviorAction

Unsupported properties to be hidden:

- visibility — not interesting in a *UML Light* context and not shown on other nodes
- isAtomic — encapsulated classifiers and components are not supported in *UML Light*
- local precondition/postcondition — the **New Child** menu is required to exclude `Constraint` types on activity elements, including `CallBehaviorAction`, so it doesn't make sense to present these here. Or, they could be moved to a **Constraints** tab as for `Interaction` and other behaviors

CallBehaviorAction

AcceptEventAction

Note that it is not clear that this element should be supported at all in the diagram.

Unsupported properties to be hidden:

- visibility — not interesting in a *UML Light* context and not shown on other nodes
- local precondition/postcondition — the **New Child** menu is required to exclude `Constraint` types on activity elements, including `AcceptEventAction`, so it doesn't make sense to present these here. Or, they could be moved to a **Constraints** tab as for `Interaction` and other behaviors

AcceptEventAction

InputPin

Unsupported properties to be hidden:

- visibility
- isControl
- isControlType
- selection — the **New Child** menu is required to exclude `Behavior` types on activity elements, including `InputPin`, so it doesn't make sense to present this here
- upperBound — this is redundant with the multiplicity Property
- isStatic

As for multiplicities in the class diagram, the editor switching button should be suppressed to prefer simply textual multiplicity specifications over exposure of the value-specification models of the bounds.

InputPin

OutputPin

Unsupported properties to be hidden:

- visibility
- isControl
- isControlType
- selection — the **New Child** menu is required to exclude `Behavior` types on activity elements, including `OutputPin`, so it doesn't make sense to present this here
- upperBound — this is redundant with the multiplicity Property
- isStatic

As for multiplicities in the class diagram, the editor switching button should be suppressed to prefer simply textual multiplicity specifications over exposure of the value-specification models of the bounds.

OutputPin

ValuePin

Unsupported properties to be hidden:

- visibility
- isControl
- isControlType
- selection — the **New Child** menu is required to exclude `Behavior` types on activity elements, including `OutputPin`, so it doesn't make sense to present this here
- upperBound — this is redundant with the multiplicity Property
- isStatic

As for multiplicities in the class diagram, the editor switching button should be suppressed to prefer simply textual multiplicity specifications over exposure of the value-specification models of the bounds.

ValuePin